```
// SysTick.c
// Runs on LM4F120/TM4C123
// 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
// September 11, 2013

/* This example accompanies the books
   "Embedded Systems: Introduction to ARM Cortex M Microcontrollers",
   ISBN: 978-1469998749, Jonathan Valvano, copyright (c) 2014
   Volume 1, Program 4.7

   "Embedded Systems: Real Time Interfacing to ARM Cortex M Microcontrollers",
   ISBN: 978-1463590154, Jonathan Valvano, copyright (c) 2014
   Program 2.11, Section 2.6

 Copyright 2014 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/
 */

#include <stdint.h>
#include "tm4c123gh6pm.h"
#define NVIC_ST_CTRL_COUNT      0x00010000  // Count flag
#define NVIC_ST_CTRL_CLK_SRC    0x00000004  // Clock Source
#define NVIC_ST_CTRL_INTEN      0x00000002  // Interrupt enable
#define NVIC_ST_CTRL_ENABLE     0x00000001  // Counter mode
#define NVIC_ST_RELOAD_M        0x00FFFFFF  // Counter load value

// Initialize SysTick with busy wait running at bus clock.
void SysTick_Init(void){
  NVIC_ST_CTRL_R = 0;                   // disable SysTick during setup
  NVIC_ST_RELOAD_R = NVIC_ST_RELOAD_M;  // maximum reload value
  NVIC_ST_CURRENT_R = 0;                // any write to current clears it
                                        // enable SysTick with core clock
  NVIC_ST_CTRL_R = NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC;
}
// Time delay using busy wait.
// The delay parameter is in units of the core clock. (units of 20 nsec for 50 MHz clock)
void SysTick_Wait(uint32_t delay){
  volatile uint32_t elapsedTime;
  uint32_t startTime = NVIC_ST_CURRENT_R;
  do{
    elapsedTime = (startTime-NVIC_ST_CURRENT_R)&0x00FFFFFF;
  }
  while(elapsedTime <= delay);
}
// Time delay using busy wait.
// This assumes 50 MHz system clock.
void SysTick_Wait10ms(uint32_t delay){
  uint32_t i;
  for(i=0; i<delay; i++){
    SysTick_Wait(800000);  // wait 10ms (assumes 50 MHz clock)
  }
}
```

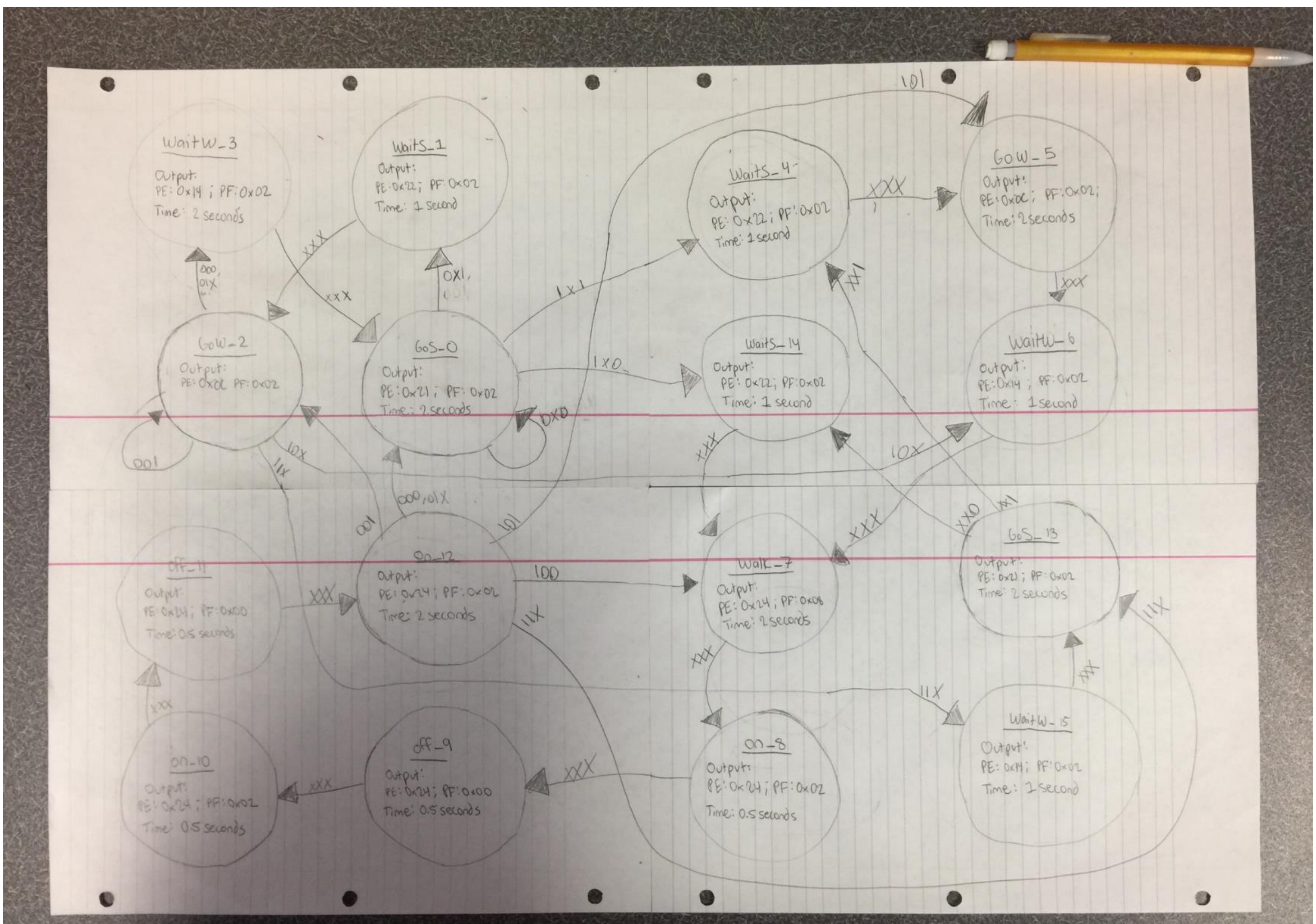
```
// TableTrafficLight.c solution to edX lab 10, EE319KLab 5
// Runs on LM4F120 or TM4C123
// Index implementation of a Moore finite state machine to operate a traffic light.
// Daniel Valvano, Jonathan Valvano
// November 7, 2013

/* solution, do not post

 Copyright 2014 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/west red light connected to PE5
// east/west yellow light connected to PE4
// east/west green light connected to PE3
// north/south facing red light connected to PE2
// north/south facing yellow light connected to PE1
// north/south facing green light connected to PE0
// pedestrian detector connected to PB2 (1=pedestrian present)
// north/south car detector connected to PB1 (1=car present)
// east/west car detector connected to PB0 (1=car present)
// "walk" light connected to PF3 (built-in green LED)
// "don't walk" light connected to PF1 (built-in red LED)
// Modified by: Michael Hernandez and Arkan Abuyazid
// Last Modified: 3/8/2017
#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "SysTick.h"
#include "TExaS.h"
#define goS_0    0
#define waitS_1  1
#define goW_2   2
#define waitW_3  3
#define waitS_4  4
#define goW_5    5
#define waitW_6  6
#define walk_7   7
#define on_8     8
#define off_9    9
#define on_10    10
#define off_11   11
#define on_12    12
#define goS_13   13
#define waitS_14 14
#define waitW_15 15
uint8_t input;
uint8_t CS;

// Declare your FSM linked structure here
```

```
struct State{

	//first element of output contains output for Port E
	//second element of output contains output for Port F

	uint8_t output[2];

	//ascertains next state

	uint8_t next[8];

	//contains wait time

	uint16_t wait;

};

struct State FSM[16] = {

	{{0x21, 0x02}, {goS_0, waitS_1, goS_0, waitS_1, waitS_14, waitS_4, waitS_14, waitS_4}, 200},
								//goS_0

	{{0x22, 0x02}, {goW_2, goW_2, goW_2, goW_2, goW_2, goW_2,goW_2, goW_2}, 100},

	//waitS_1

	{{0x0C, 0x02}, {waitW_3, goW_2, waitW_3, waitW_3, waitW_6, waitW_6, waitW_15,
waitW_15}, 200},								//goW_2
```

```
        {{0x14, 0x02}, {goS_0, goS_0, goS_0, goS_0, goS_0, goS_0, goS_0, goS_0}, 100},
                                                                                            //waitW_3
CHANGED

        {{0x22, 0x02}, {goW_5, goW_5, goW_5, goW_5, goW_5, goW_5, goW_5, goW_5}, 100},

        //waitS_4

        {{0x0C, 0x02}, {waitW_6, waitW_6, waitW_6, waitW_6, waitW_6, waitW_6, waitW_6, waitW_6},
200},                                   //goW_5

        {{0x14, 0x02}, {walk_7, walk_7, walk_7, walk_7, walk_7, walk_7, walk_7, walk_7}, 100},
                                                                        //waitW_6   CHANGED

        {{0x24, 0x08}, {on_8, on_8, on_8, on_8, on_8, on_8, on_8, on_8}, 200},

        //walk_7

        {{0x24, 0x02}, {off_9, off_9, off_9, off_9, off_9, off_9, off_9, off_9}, 50},
                                                                                            //on_8

        {{0x24, 0x00}, {on_10, on_10, on_10, on_10, on_10, on_10, on_10, on_10, }, 50},
                                                                                                   //off_9

        {{0x24, 0x02}, {off_11, off_11, off_11, off_11, off_11, off_11, off_11, off_11}, 50},
                                                                              //on_10

        {{0x24, 0x00}, {on_12, on_12, on_12, on_12, on_12, on_12, on_12, on_12}, 50},

        //off_11

        {{0x24, 0x02}, {goS_0, goW_2, goS_0, goS_0, walk_7, goW_5, goS_13, goS_13}, 200},
                                                                                            //on_12
```

```
		{{0x21, 0x02}, {waitS_14, waitS_4, waitS_14, waitS_4, waitS_14, waitS_4, waitS_14, waitS_4}, 
200},					//goS_13


		{{0x22, 0x02}, {walk_7,walk_7,walk_7,walk_7,walk_7,walk_7,walk_7, walk_7}, 100},
											//waitS_14


		{{0x14, 0x02}, {goS_13, goS_13, goS_13, goS_13, goS_13, goS_13, goS_13, goS_13}, 100}
									//waitW_15 // 18 is PE4,3 and 
this shouldn't be a combo.


};
void EnableInterrupts(void);
void SystemInit(void);
int main(void){ //volatile unsigned long delay;


		// activate traffic simulation and set system clock to 80 MHz
  TExaS_Init(SW_PIN_PB210, LED_PIN_PE543210);


  SysTick_Init();


  EnableInterrupts();


  SystemInit();


  //FSM Engine


CS =goS_0;


  while(1){
```

```
                        GPIO_PORTE_DATA_R = FSM [CS].output [0];
                GPIO_PORTF_DATA_R = FSM [CS].output [1];
                SysTick_Wait10ms(FSM[CS].wait);
                input = (GPIO_PORTB_DATA_R & 0x07);
                CS = FSM[CS].next[input];

 }
}

void SystemInit(void){
        volatile uint8_t delay;

        SYSCTL_RCGC2_R |= 0x32;
        delay = 10;
        GPIO_PORTB_DIR_R &= 0xF8;
        GPIO_PORTB_DEN_R |= 0x07;
        GPIO_PORTE_DIR_R |= 0x3F;
        GPIO_PORTE_DEN_R |= 0x3F;
        GPIO_PORTF_PUR_R |= 0x0A;
        GPIO_PORTF_DIR_R |= 0x0A;
 GPIO_PORTF_DEN_R |= 0x0A;

        return;

}
```


WaitW_3
Output:
PE: 0x14 ; PF: 0x02
Time: 2 seconds
WaitS_1
Output:
PE: 0x22 ; PF: 0x02
Time: 1 second
WaitS_4
Output:
PE: 0x22 ; PF: 0x02
Time: 1 second
GoW_5
Output:
PE: 0x0C ; PF: 0x02 ;
Time: 2 seconds
GoW_2
Output:
PE: 0x0C PF: 0x02
GoS_0
Output:
PE: 0x21 ; PF: 0x02
Time: 2 seconds
WaitS_14
Output:
PE: 0x22 ; PF: 0x02
Time: 1 second
WaitW_6
Output:
PE: 0x14 ; PF: 0x02
Time: 1 second
off_11
Output:
PE: 0x24 ; PF: 0x00
Time: 0.5 seconds
on_12
Output:
PE: 0x24 ; PF: 0x02
Time: 2 seconds
Walk_7
Output:
PE: 0x24 ; PF: 0x08
Time: 2 seconds
GoS_13
Output:
PE: 0x21 ; PF: 0x02
Time: 2 seconds
on_10
Output:
PE: 0x24 ; PF: 0x02
Time: 0.5 seconds
off_9
Output:
PE: 0x24 ; PF: 0x00
Time: 0.5 seconds
on_8
Output:
PE: 0x24 ; PF: 0x02
Time: 0.5 seconds
WaitW_15
Output:
PE: 0x14 ; PF: 0x02
Time: 1 second
101
XXX
XXX
xxx
000, 01X
0X1,
1x1
1X0
0X0
001
10X
11X
000, 01X
001
101
XX1
10X
XXX
XXX
XX0
XX1
XXX
100
11X
XXX
11X
XXX
XXX
XXX
11X