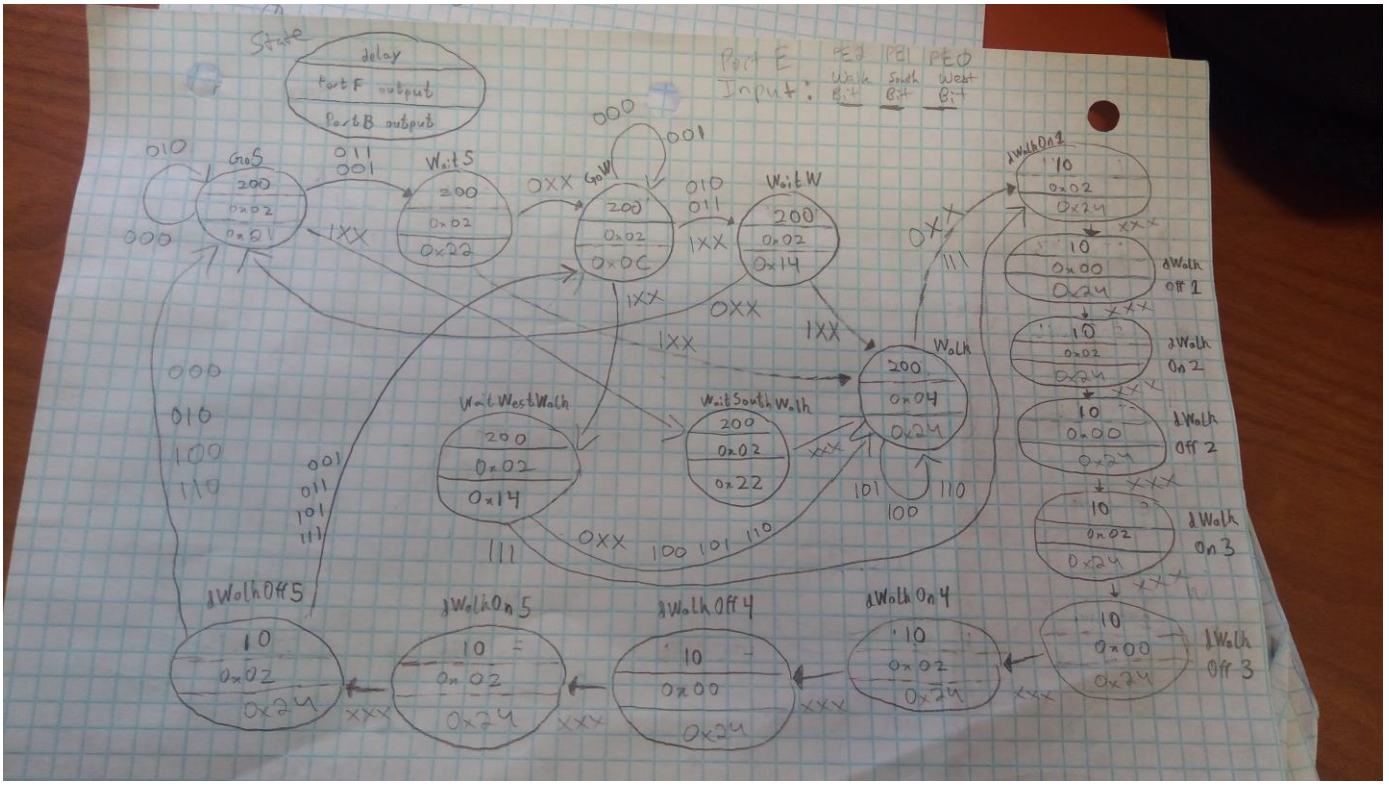

// TableTrafficLight.c solution to edX lab 10, EE319KLab 5

// 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

```
// 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)
#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "SysTick.h"
#include "TExaS.h"

// Declare your FSM linked structure here
```

```
struct State {
        int8_t Bout;      // Port B output
        int8_t Fout;      // Port F output
        int16_t delay;    //
        int next[8];
};

typedef const struct State StateType;

        StateType Fsm[17] = {
                {0x21, 0x02, 200, {0,1,0,1,15,15,15,15}}, // 0. GoSouth
                {0x22, 0x02, 200, {2,2,2,2,4,4,4,4}},     // 1. waitSouth
                {0x0C, 0x02, 200, {2,2,3,3,16,16,16,16}}, // 2. GoWest
                {0x14, 0x02, 200, {0,0,0,0,4,4,4,4}},     // 3. waitWest
                {0x24, 0x08, 200, {5,5,5,5,4,4,4,5}},     // 4. walk
                {0x24, 0x02, 10, {6,6,6,6,6,6,6,6}},      // 5. dwalkOn1
                {0x24, 0x00, 10, {7,7,7,7,7,7,7,7}},      // 6. dwalkOff1
                {0x24, 0x02, 10, {8,8,8,8,8,8,8,8}},      // 7. dwalkOn2
                {0x24, 0x00, 10, {9,9,9,9,9,9,9,9}},      // 8. dwalkOff2
                {0x24, 0x02, 10, {10,10,10,10,10,10,10,10}},    // 9. dwalkOn3
                {0x24, 0x00, 10, {11,11,11,11,11,11,11,11}},    // 10. dwalkOff3
                {0x24, 0x02, 10, {12,12,12,12,12,12,12,12}},    // 11. dwalkOn4
                {0x24, 0x00, 10, {13,13,13,13,13,13,13,13}},    // 12. dwalkOff4
                {0x24, 0x02, 10, {14,14,14,14,14,14,14,14}},    // 13. dwalkOn5
                {0x24, 0x00, 10, {0,2,0,2,0,2,0,2}},            // 14. dwalkOff5
                {0x22, 0x02, 200, {4,4,4,4,4,4,4,4}},     // 15. waitSouthWalk
                {0x14, 0x02, 200, {4,4,4,4,4,4,4,0}},     // 16. waitWestWalk
        };
```

```
void EnableInterrupts(void);


int main(void){ volatile unsigned long delay;
        int cstate = 0;
        int input;
        TExaS_Init(SW_PIN_PE210, LED_PIN_PB543210); // activate traffic simulation and set system clock to 80 MHz
 SysTick_Init();
 EnableInterrupts();
        // turn on clock
        SYSCTL_RCGC2_R |= 0x00000032;
        delay = SYSCTL_RCGCGPIO_R;
        // Port F Init
 GPIO_PORTF_DIR_R |= 0x0A;
 GPIO_PORTF_DEN_R |= 0x0A;
        GPIO_PORTF_AFSEL_R = 0x00;
        // Port B Init
 GPIO_PORTB_AFSEL_R = 0x00;
 GPIO_PORTB_DIR_R |= 0x3F;
 GPIO_PORTB_DEN_R |= 0x3F;
        // Port E Init
 GPIO_PORTE_DIR_R |= 0x00;
 GPIO_PORTE_AFSEL_R = 0x00;
 GPIO_PORTE_DEN_R |= 0x007;


 //FSM Engine
 while(1){
                GPIO_PORTB_DATA_R = Fsm[cstate].Bout;        // Port B output
                GPIO_PORTF_DATA_R = Fsm[cstate].Fout;        // Port F output
```

```
                SysTick_Wait10ms((Fsm[cstate].delay)); // delay
                input = (int) GPIO_PORTE_DATA_R & 0x07;        // button input
                cstate = Fsm[cstate].next[input];            // update state
 }
}
```

```
// SysTick.c
// Implements two busy-wait based delay routines
#include <stdint.h>
// 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))
#define reload 0x00FFFFFF
void SysTick_Init(void){
        NVIC_ST_CTRL_R = 0;
        NVIC_ST_RELOAD_R = reload;
        NVIC_ST_CTRL_R = 0x00000005;
}
```

```
// The delay parameter is in units of the 80 MHz core clock. (12.5 ns)
void SysTick_Wait(unsigned long delay){                    //cycles * 12.5e^-9 = delay time in sec
        NVIC_ST_RELOAD_R = delay - 1;
        NVIC_ST_CURRENT_R = 0;
        while ((NVIC_ST_CTRL_R & 0x00010000)==0){ // bit 16 has 1 if Systick_wait has reached 0
        }
                                                           // once it has reched zero, it stops looping
}

// Time delay using busy wait.
// waits for count*10ms
// 10000us equals 10ms
void SysTick_Wait10ms(uint32_t delay){
        uint32_t i;
        for (i = 0;i<delay;i++){
                SysTick_Wait(800000);                      //12.5 ns times 800,000 = 10 ms
        }

}
```