Lab 4 Report

Kerim Turak

Video: https://youtu.be/ez-_NQETJ_Y

a. Ports Initialization:

I named ALL the necessary registers for ports initialization. Do it carefully. **Activate the clock**, then allow time for the clock to start. **Unlock GPIO** for each port by the magic number 0x4C4F434B, then **allow changes to ports** and bits you use by setting the committed register. **Disable the analog** functionality since we won't use it. **Disable the PCTL** register. **Config the direction register** for I/O (Input = 0 and Output = 1). **Disable alternate function** and **pull-up resistors**. Finally, **enable digital** I/O for ports and bits we are going to use.
Important, remember to set PCTL = 0. I mistakenly set PCTL (port control) to be equal to another number, and the automatic grading machine aborted my program.

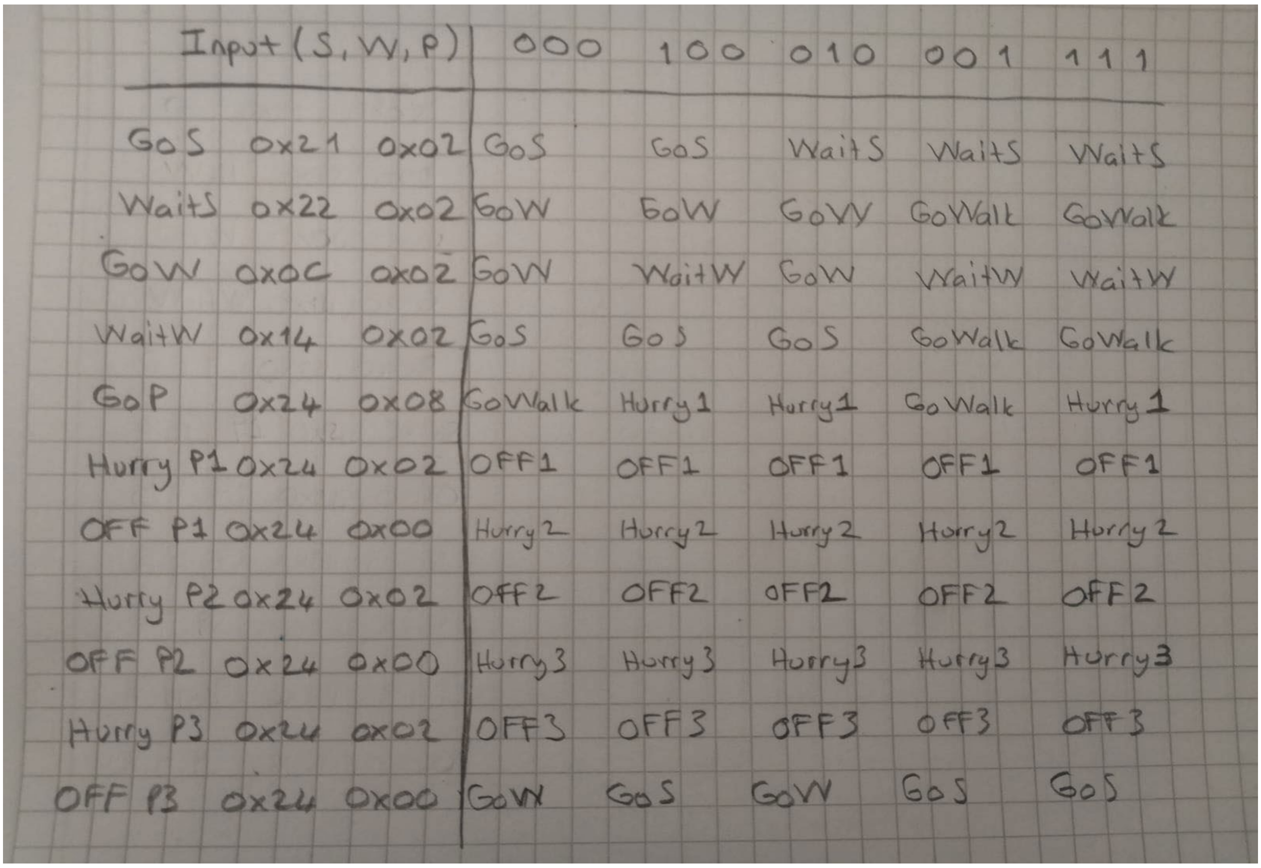
b. FSM declaration:

The struct must have at least 4 elements, and we must output to the roads before indicating anything on the pedestrian. Every state must have a wait time. The last element, also the most important construction of an FSM, is state transition array. Draw a table with possible inputs, then write possible outputs based on the input and the current state. For references, I made this table below. Remember that **the FSM declaration and the table are EXACTLY the same things**.

FINITE-STATE MACHINES (FSM):

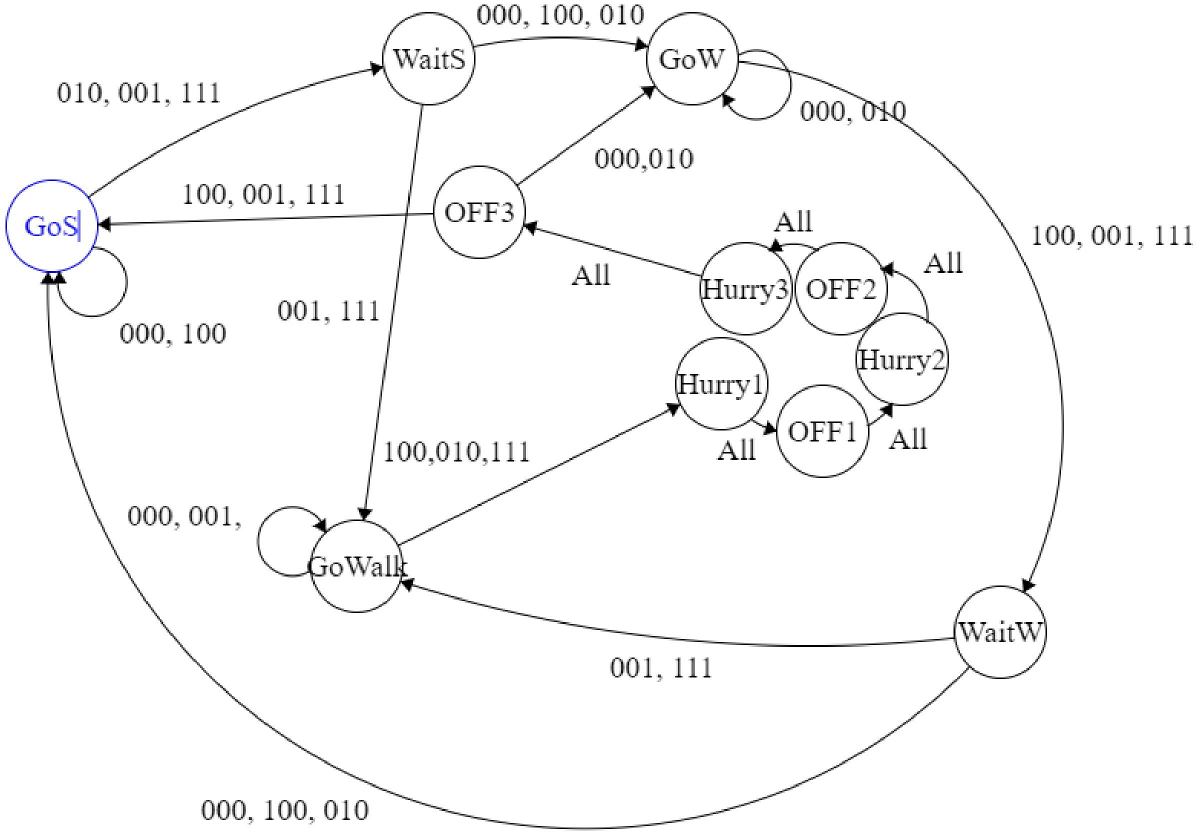
The most abstract content in this chapter is FSM. To fully describe an FSM, we need **5 things**:

1. Set of inputs
2. Set of outputs
3. Set of states
4. State transition graphs or matrix
5. Output determination



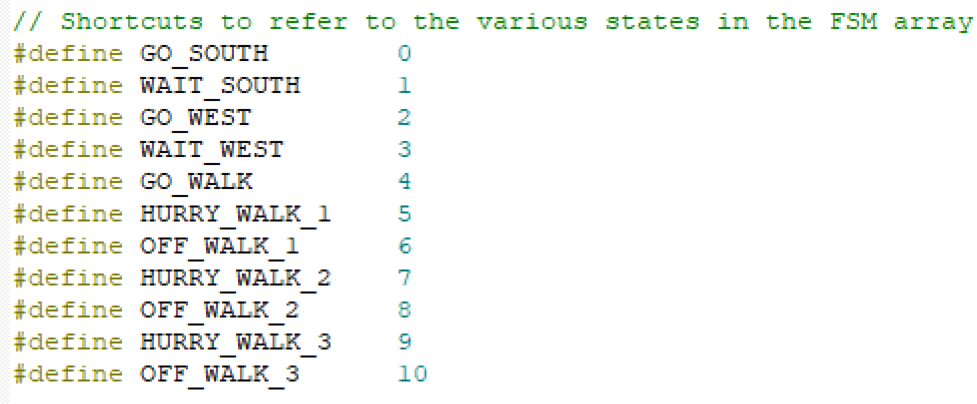
Input (S, W, P)			000	100	010	001	111
GoS	0x21	0x02	GoS	GoS	WaitS	WaitS	WaitS
WaitS	0x22	0x02	GoW	GoW	GoW	GoWalk	GoWalk
GoW	0x0C	0x02	GoW	WaitW	GoW	WaitW	WaitW
WaitW	0x14	0x02	GoS	GoS	GoS	GoWalk	GoWalk
GoP	0x24	0x08	GoWalk	Hurry1	Hurry1	GoWalk	Hurry1
Hurry P1	0x24	0x02	OFF1	OFF1	OFF1	OFF1	OFF1
OFF P1	0x24	0x00	Hurry2	Hurry2	Hurry2	Hurry2	Hurry2
Hurry P2	0x24	0x02	OFF2	OFF2	OFF2	OFF2	OFF2
OFF P2	0x24	0x00	Hurry3	Hurry3	Hurry3	Hurry3	Hurry3
Hurry P3	0x24	0x02	OFF3	OFF3	OFF3	OFF3	OFF3
OFF P3	0x24	0x00	GoW	GoS	GoW	GoS	GoS

The order of the code segments in `while (1)` infinite looping must be:
roads outputs ~> pedestrian outputs ~> wait ~> get inputs ~> state transition



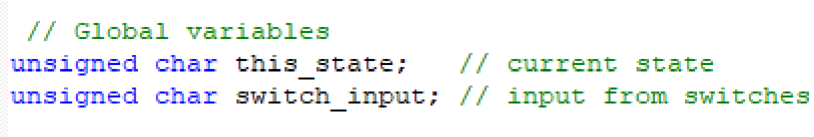
Explanation

To use delay I use delay function which we write previous labs. I write the addresses of the ports, that we use and by writen a function called the Portsinit I assign the needed values to registers and this function is called in main function to initialize the ports. I also determined the all state to complete the algorithm as below.



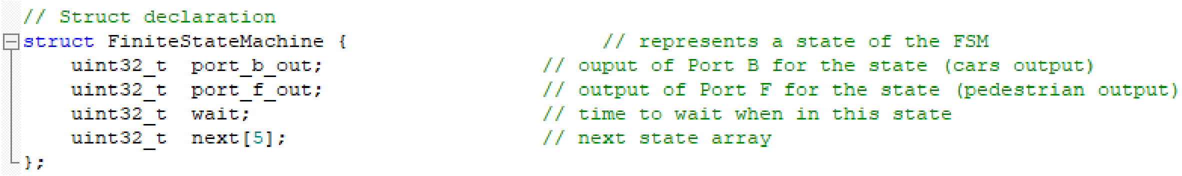
```
// Shortcuts to refer to the various states in the FSM array
#define GO_SOUTH        0
#define WAIT_SOUTH      1
#define GO_WEST         2
#define WAIT_WEST       3
#define GO_WALK         4
#define HURRY_WALK_1    5
#define OFF_WALK_1      6
#define HURRY_WALK_2    7
#define OFF_WALK_2      8
#define HURRY_WALK_3    9
#define OFF_WALK_3      10
```

After this operations, the variables are declared to control input and the state as below.



```
 // Global variables
unsigned char this_state;   // current state
unsigned char switch_input; // input from switches
```

I searched the struct to complete the code because of I saw some research used it, as below.



```
// Struct declaration
struct FiniteStateMachine {                        // represents a state of the FSM
    uint32_t  port_b_out;                      // ouput of Port B for the state (cars output)
    uint32_t  port_f_out;                      // output of Port F for the state (pedestrian output)
    uint32_t  wait;                            // time to wait when in this state
    uint32_t  next[5];                         // next state array
};
```

Source: https://www.tutorialspoint.com/cprogramming/c_structures.htm

After the struct all code got easy.

MY CODE

```
/**
 * Required hardware I/O connections
 *
 * West's Red-Yellow-Green connected to PB5-PB4-PB3
 * South's Red-Yellow-Green connected to  PB2-PB1-PB0
 * Pedestrian's Red connected to PF1
 * Pedestrian's Green connected to PF3
 * West's switch connected to PE0
 * South's switch connected to PE1
 * Pedestrian's switch connected to PE2 PF4
 */
#include <stdint.h>
#include <stdlib.h>

// Port F
#define GPIO_PORTF_DATA_R     (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R      (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_AFSEL_R    (*((volatile unsigned long *)0x40025420))
#define GPIO_PORTF_PUR_R      (*((volatile unsigned long *)0x40025510))
#define GPIO_PORTF_DEN_R      (*((volatile unsigned long *)0x4002551C))
#define GPIO_PORTF_LOCK_R     (*((volatile unsigned long *)0x40025520))
#define GPIO_PORTF_CR_R       (*((volatile unsigned long *)0x40025524))
#define GPIO_PORTF_AMSEL_R    (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R     (*((volatile unsigned long *)0x4002552C))
// Port B
#define GPIO_PORTB_DATA_R     (*((volatile unsigned long *)0x400053FC))
#define GPIO_PORTB_DIR_R      (*((volatile unsigned long *)0x40005400))
#define GPIO_PORTB_AFSEL_R    (*((volatile unsigned long *)0x40005420))
#define GPIO_PORTB_PUR_R      (*((volatile unsigned long *)0x40005510))
#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))
// Port E
#define GPIO_PORTE_DATA_R     (*((volatile unsigned long *)0x400243FC))
#define GPIO_PORTE_DIR_R      (*((volatile unsigned long *)0x40024400))
#define GPIO_PORTE_AFSEL_R    (*((volatile unsigned long *)0x40024420))
#define GPIO_PORTE_PUR_R      (*((volatile unsigned long *)0x40024510))
#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))
// System Clock
#define SYSCTL_RCGC2_R        (*((volatile unsigned long *)0x400FE108))

// Shortcuts to refer to the various states in the FSM array
#define GO_SOUTH       0
#define WAIT_SOUTH     1
#define GO_WEST        2
#define WAIT_WEST      3
#define GO_WALK        4
#define HURRY_WALK_1   5
#define OFF_WALK_1     6
#define HURRY_WALK_2   7
#define OFF_WALK_2     8
#define HURRY_WALK_3   9
#define OFF_WALK_3     10

 // Global variables
unsigned char this_state;   // current state
unsigned char switch_input; // input from switches

// Struct declaration
```

```
struct FiniteStateMachine {                  // represents a state of the FSM
   uint32_t  port_b_out;                // ouput of Port B for the state (cars output)
   uint32_t  port_f_out;                // output of Port F for the state (pedestrian output)
   uint32_t  wait;                     // time to wait when in this state
   uint32_t  next[5];                   // next state array
};
 typedef const struct FiniteStateMachine STATE;
// FSM declaration
STATE FSM[11] = {
   // 0) Go South
   {0x21, 0x02, 30,{ GO_SOUTH, GO_SOUTH, WAIT_SOUTH, WAIT_SOUTH, WAIT_SOUTH }},
   // 1) Wait South
   {0x22, 0x02,  5,{ GO_WEST, GO_WEST, GO_WEST, GO_WALK, GO_WEST }},
   // 2) Go West
   {0x0C, 0x02, 30,{ GO_WEST, WAIT_WEST, GO_WEST, WAIT_WEST, WAIT_WEST }},
   // 3) Wait West
   {0x14, 0x02,  5,{ GO_SOUTH, GO_SOUTH, GO_SOUTH, GO_WALK, GO_WALK }},
   // 4) Go Pedestrian
   {0x24, 0x08, 30,{ GO_WALK, HURRY_WALK_1, HURRY_WALK_1, GO_WALK, HURRY_WALK_1 }},
   // 5) Hurry Pedestrian 1
   {0x24, 0x02,  2,{ OFF_WALK_1, OFF_WALK_1, OFF_WALK_1, OFF_WALK_1, OFF_WALK_1 }},
   // 6) Off Pedestrian 1
   {0x24, 0x00,  2,{ HURRY_WALK_2, HURRY_WALK_2, HURRY_WALK_2, HURRY_WALK_2,
HURRY_WALK_2 }},
   // 7) Hurry Pedestrian 2
   {0x24, 0x02,  2,{ OFF_WALK_2, OFF_WALK_2, OFF_WALK_2, OFF_WALK_2, OFF_WALK_2 }},
   // 8) Off Pedestrian 2
   {0x24, 0x00,  2,{ HURRY_WALK_3, HURRY_WALK_3, HURRY_WALK_3, HURRY_WALK_3,
HURRY_WALK_3 }},
   // 9) Hurry Pedestrian 3:
   {0x24, 0x02,  2,{ OFF_WALK_3, OFF_WALK_3, OFF_WALK_3, OFF_WALK_3, OFF_WALK_3 }},
   // 10) Off Pedestrian 3:
```

```
  {0x24, 0x00,  2,{ GO_WEST, GO_SOUTH, GO_WEST, GO_SOUTH,  GO_SOUTH }}
};
void PortsInit(void) {
  // 1) activate clock for Port F, Port B, and Port E
  SYSCTL_RCGC2_R |= 0x00000032;
  // Port F
  GPIO_PORTF_LOCK_R  = 0x4C4F434B;  // 2) unlock GPIO Port F
  GPIO_PORTF_CR_R   |= 0x0A;       // allow changes to PF3, PF1
  GPIO_PORTF_AMSEL_R = 0x00;       // 3) disable analog function
  GPIO_PORTF_PCTL_R  = 0x00;       // 4) PCTL GPIO on PF3, PF1
  GPIO_PORTF_DIR_R  |= 0x0A;       // 5) PF3, PF1 are outputs
  GPIO_PORTF_AFSEL_R = 0x00;       // 6) disable alternate function
  GPIO_PORTF_PUR_R   = 0x00;       // disable pull-up resistor
  GPIO_PORTF_DEN_R  |= 0x0A;       // 7) enable digital I/O on PF3, PF1
  // Port B
  GPIO_PORTB_AMSEL_R = 0x00;       // 3) disable analog function
  GPIO_PORTB_PCTL_R  = 0x00;       // 4) PCTL GPIO on PB5-PB0
  GPIO_PORTB_DIR_R  |= 0x3F;       // 5) PB5-PB0 are outputs
  GPIO_PORTB_AFSEL_R = 0x00;       // 6) disable alternate function
  GPIO_PORTB_PUR_R   = 0x00;       // disable pull-up resistor
  GPIO_PORTB_DEN_R  |= 0x3F;       // 7) enable digital I/O on PB5-PB0
  // Port E
  GPIO_PORTE_AMSEL_R = 0x00;       // 3) disable analog function
  GPIO_PORTE_PCTL_R  = 0x00;       // 4) PCTL GPIO on PE2-PE0
  GPIO_PORTE_DIR_R   = 0x00;       // 5) PE2-PE0 are inputs
  GPIO_PORTE_AFSEL_R = 0x00;       // 6) disable alternate function
  GPIO_PORTE_PUR_R   = 0x00;       // disable pull-up resistor
  GPIO_PORTE_DEN_R  |= 0x07;       // 7) enable digital I/O on PE2-PE0
}
// delay function
void delay(int sec){
        int c = 1, d = 1;
        for( c = 1; c <= sec; c++)
                for( d = 1; d <= 400000; d++){}//1
}

int  main(void) {
  PortsInit();
  while (1) {
                          // make outputs
    GPIO_PORTB_DATA_R  = FSM[this_state].port_b_out; // to cars (port B)
    GPIO_PORTF_DATA_R  = FSM[this_state].port_f_out; // to pedestrians (port F)
    delay(FSM[this_state].wait);

    // get inputs
    // if no switch is pressed
    if (GPIO_PORTE_DATA_R  == 0x00) {
      switch_input = 0; // then it is case 0 of the next[] array...
    }  // ... all LEDs stay the way they are since the last pressing
     // if south switch is pressed
    else if (GPIO_PORTE_DATA_R == 0x02) {
      switch_input = 1; // then it is case 1 of the next[] array...
    }  // ... all LEDs correspond to Go South mode
     // if west switch is pressed
    else if (GPIO_PORTE_DATA_R == 0x01) {
      switch_input = 2; // then it is case 2 of the next[] array...
    }  // ... all LEDs correspond to Go West mode
     // if pedestrian switch is pressed
    else if (GPIO_PORTE_DATA_R == 0x04) {
      switch_input = 3; // then it is case 3 of the next[] array...
    }  // ... all LEDs correspond to Go Pedestrian mode
     // if all switches are pressed
```

```
        else if (GPIO_PORTE_DATA_R == 0x07) {
            switch_input = 4; // then it is case 4 of the next[] array...
        }  // ... all LEDs correspond periodically: South, West, Pedestrian
           // change state based on input and current state
        this_state = FSM[this_state].next[switch_input];
    }
}
```