Lab#03

Traffic Light Controller using FSM

By
Hitesh Vijay Oswal
Ankit Vilasrao Komawar

EECE 544: Digital Systems Design

Date: 10/15/2015

I. INTRODUCTION

1. Objective:

To develop a Traffic Light Controller for 2 – 2 way streets using Finite State Machine(FSM) in C Language on Tiva Board. The System has 5 inputs in total, out of which 4 inputs represents walk signal request and 1 input represents Side Street signal request. Main Street has highest priority.
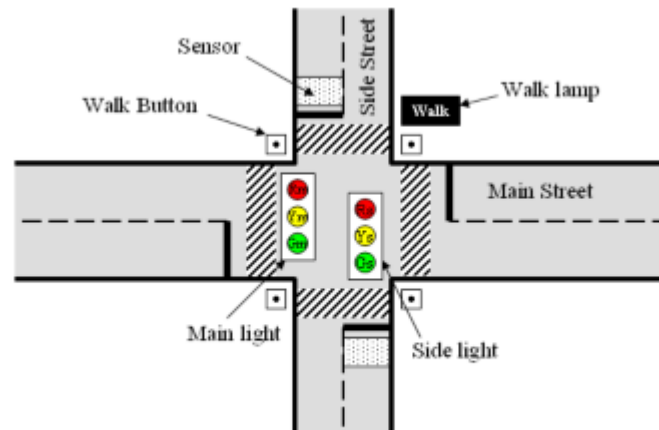

Figure 1: Diagram for intersection with corresponding lights.

Figure#1[1]: Traffic Signal System

2. Basic Theory:

The system is designed using C language. Finite State Machine (FSM) is used to design the system. The Delay between to states is provided using SysTick. Linked List Structure is used in C Language to represent single State. A Structure includes Outputs to be given on 2 ports, delay of the present state and pointer array of next states depending upon the inputs. A pointer is used to point the present state. This pointer is updated to change the state.

The controller used in the Tiva board is TM4C123G. This controller works on 80MHz clock frequency, which is obtained using PLL with the input frequency of 16MHz. It has 6 Ports and verity of different functions. The development tool used is Keil µVision MDK-ARM.

II. BODY

1. Hardware Connections:

In this system 3 Ports are used and they are, Port B, Port E and Port F.

i.   **Port B:** In this there 3 PINs are configured as output PINs. LEDs are used to represent outputs. All LEDs are connected in active high configuration. PB-3 represents Green light for Side Street, PB-4 represents Yellow light for Side Street and PB-5 represents Red light for Side Street.

ii. **Port E:** In this there 4 PINs are configured as output PINs and 2 PINs are configured as Input PINs. LEDs are used to represent outputs and normally open switches are used to get inputs. All LEDs are connected in active high configuration and Switches are connected in Negative logic configuration. PE-

3 represents Green light for Main Street, PE-4 represents Yellow light for Main Street and PE-5 represents Red light for Main Street. PE-2 represents Green light for Walk Signal. 2 walk signal request inputs are given on PE-1 and PE-0. These PINs have active Internal Pull-up resistors.

iii. **Port F:** In this there 3 PINs are configured as Input PINs. Normally open switches are used to get inputs. All the switches are connected in Negative logic configuration. All the 3 PINs have active Internal Pull-up resistors. PINs PF-0 and PF-1 are used for walk signal request inputs and PF-2 is used for Side Street Signal Request (Sensor).

2. PLL Configuration[2]:

XTAL	Crystal Freq (MHz)	XTAL	Crystal Freq (MHz)
0x0	Reserved	0x10	10.0 MHz
0x1	Reserved	0x11	12.0 MHz
0x2	Reserved	0x12	12.288 MHz
0x3	Reserved	0x13	13.56 MHz
0x4	3.579545 MHz	0x14	14.31818 MHz
0x5	3.6864 MHz	0x15	16.0 MHz
0x6	4 MHz	0x16	16.384 MHz
0x7	4.096 MHz	0x17	18.0 MHz
0x8	4.9152 MHz	0x18	20.0 MHz
0x9	5 MHz	0x19	24.0 MHz
0xA	5.12 MHz	0x1A	25.0 MHz
0xB	6 MHz (reset value)	0x1B	Reserved
0xC	6.144 MHz	0x1C	Reserved
0xD	7.3728 MHz	0x1D	Reserved
0xE	8 MHz	0x1E	Reserved
0xF	8.192 MHz	0x1F	Reserved

Table 4.9a. XTAL field used in the SYSCTL_RCC_R register of the TM4C123.

Table#1: XTAL Field used in the SYSCTL_RCC_R

Address	26-23	22	13	11	10-6	5-4	Name
$400FE060	SYSDIV	USESYSDIV	PWRDN	BYPASS	XTAL	OSCSRC	SYSCTL_RCC_R
$400FE050					PLLRIS		SYSCTL_RIS_R
	31	30	28-22	13	11	6-4	
$400FE070	USERCC2	DIV400	SYSDIV2	PWRDN2	BYPASS2	OSCSRC2	SYSCTL_RCC2_R

Table 4.9b. Main clock registers for the TM4C123.

Table#2: Main Clock register

i. Use RCC2 because it provides for more options.
ii. The first step is set BYPASS2 (bit 11). At this point the PLL is bypassed and there is no system clock divider.
iii. The second step is to specify the crystal frequency in the four XTAL bits using the code in Table 4.9. The OSCSRC2 bits are cleared to select the main oscillator as the oscillator clock source.
iv. The third step is to clear PWRDN2 (bit 13) to activate the PLL.
v. The fourth step is to configure and enable the clock divider using the 7-bit SYSDIV2 field. If the 7-bit SYSDIV2 is n, then the clock will be divided by n + 1. To get the desired 80 MHz from the 400 MHz PLL, we need to divide by 5. So, we place a 4 into the SYSDIV2 field.
vi. The fifth step is to wait for the PLL to stabilize by waiting for PLLRIS (bit 6) in the SYSCTL_RIS_R to become high.
vii. The last step is to connect the PLL by clearing the BYPASS2 bit.

3. SysTick:

SysTick is used to provide delay between the two state transitions. It is configured as follows:

Address	31-24	23-17	16	15-3	2	1	0	Name
$E000E010	0	0	COUNT	0	CLK_SRC	INTEN	ENABLE	NVIC_ST_CTRL_R
$E000E014	0	24-bit RELOAD value						NVIC_ST_RELOAD_R
$E000E018	0	24-bit CURRENT value of SysTick counter						NVIC_ST_CURRENT_R

Table 4.10. SysTick registers.

Table#3[3]: SysTick Registers

i. Disable the timer by resetting (writing 0) Enable (Bit-0 of (NVIC_ST_CTRL_R)).
ii. Write the Reload Value to Reload Register. This value is calculated using the following formulae:

$$\text{Reload Value} = \text{Time} \times \text{Frequency}$$

iii. Clear the current register by writing any value to it.
iv. Enable the timer by setting (writing 1) Enable (Bit-0 of (NVIC_ST_CTRL_R)).

4. Finite State Machine(FSM) State Table and State Graph:

INPUT	00	01	10	11
Present State and output	Next State			
Go main 0x20, 0x08, 300 (3 Sec)	Go main	Wait Main	Wait Main	Wait Main
Wait Main 0x20, 0x10, 100 (1 Sec)	Go Side	Go Side	Walk	Walk
Go Side 0x08, 0x20, 300 (3 Sec)	Wait Side	Wait Side	Wait Side	Wait Side
Wait Side 0x10, 0x20, 100 (1 Sec)	Go main	Go main	Walk	Go main
Walk 0x20, 0x24, 400 (4 Sec)	Go main	Go Side	Go main	Go Side

Table#4: State Table

NOTE: In this Experiment a delay of 10ms is initialized and called multiple times to obtain required delay.
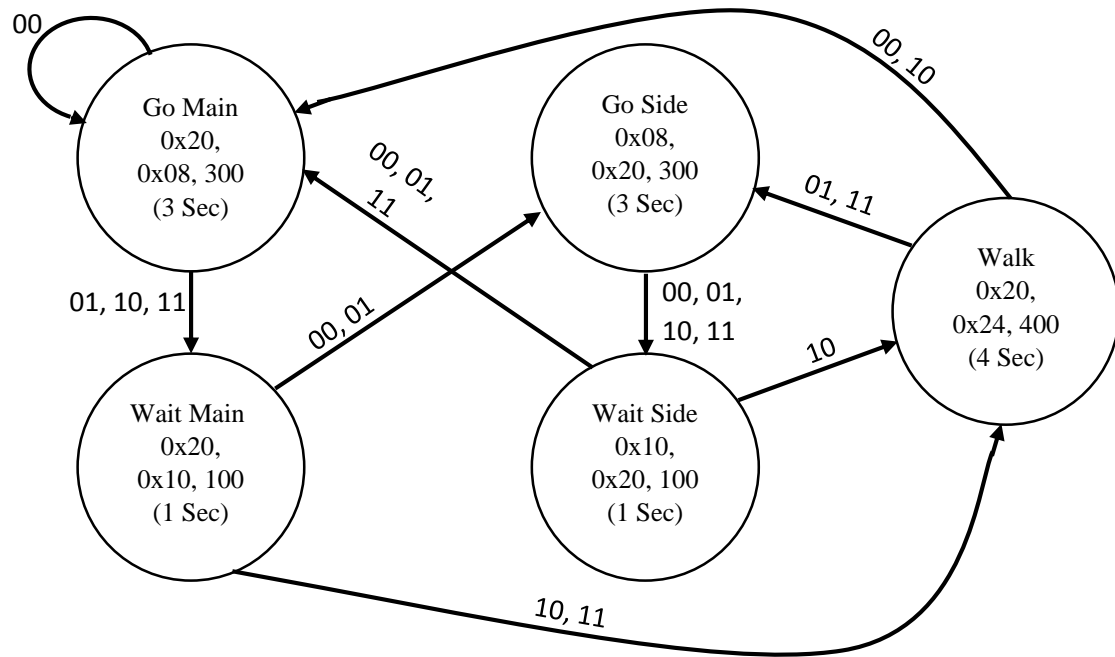
Figure#2: State Graph

- **Working of main loop:**

    i.   Initialize PLL, Port B, Port E and Port F.
    ii. Declare a Pointer of same data type as of States.
    iii. Point the pointer to Go Main State (Initial State).
    iv. Give the Outputs to the respective Ports.
    v.   Call Delay using Systick.
    vi. Check and calculate the Input.
    vii. Update the pointer to the next state according to the input.
    viii. Go to step iv.

- **Procedure to calculate input:**

    i.   Check for the all input PINs
    ii. Declare a variable INPUT and initialize it to 0x00
    iii. If any of the walk signal request button is pressed set the second bit of INPUT by ORing it with 0x02.
    iv. If Side Street signal request button is pressed (Sensor) set the first bit of INPUT by ORing it with 0x01.

- **Flow Chart:**

```
                        ┌─────────────┐
                        │    Start    │
                        └──────┬──────┘
                               ↓
                        ┌─────────────┐
                        │ Initialize PLL │
                        └──────┬──────┘
                               ↓
                        ┌─────────────┐
                        │ Initialize Port B │
                        └──────┬──────┘
                               ↓
                        ┌─────────────┐
                        │ Initialize Port E │
                        └──────┬──────┘
                               ↓
                        ┌─────────────┐
                        │ Initialize Port F │
                        └──────┬──────┘
                               ↓
                    ┌─────────────────────┐
                    │ Declare Pointer of Structure │
                    └──────────┬──────────┘
                               ↓
                    ┌─────────────────────┐
                    │ Point the pointer to Initial State │
                    └──────────┬──────────┘
                               ↓
                    ╱─────────────────────╱
                   ╱ Give the Output of    ╱
                  ╱  respective state to   ╱
                 ╱   both the Data Register╱
                 ╱───────────┬────────────╱
                             ↓
                    ┌─────────────────────┐
                    │ Give the delay according to │
                    │ present state using SysTick │
                    └──────────┬──────────┘
                               ↓
                    ┌──┬──────────────┬──┐
                    │  │ Calculate the Input │  │
                    └──┴──────┬───────┴──┘
                             ↓
                    ┌─────────────────────┐
                    │ Update the Pointer   │
                    │ According to the Input │
                    └──────────┬──────────┘
                               ↓
```

```
                    ┌─────────────────────────┐
                    │  Start Input Calculation │
                    └─────────────────────────┘
                                │
                                ▼
                    ╱─────────────────────────╱
                   ╱    Read Data Registers   ╱
                  ╱─────────────────────────╱
                                │
                                ▼
                    ┌─────────────────────────┐
                    │  Mask the Data with Input│
                    │         PINs             │
                    └─────────────────────────┘
                                │
                                ▼
                    ┌─────────────────────────┐
                    │  Declare a variable called│
                    │ Input and initialize it to 0x00│
                    └─────────────────────────┘
                                │
                                ▼
                    ┌─────────────────────────┐
                    │  Check if one or more walk│
                    │     button is pressed    │
                    └─────────────────────────┘
                                │
                                ▼
                          ╱╲
                         ╱  ╲        ┌─────────────────────────┐
                        ╱ If ╲──────▶│ Set Bit-1 of Input by ORing│
                        ╲    ╱        │    Input with 0x02       │
                         ╲  ╱         └─────────────────────────┘
                          ╲╱
                           │
                           ▼
                    ┌─────────────────────────┐
                    │ Check if side signal button is│
                    │        pressed           │
                    └─────────────────────────┘
                                │
                                ▼
                          ╱╲
                         ╱  ╲        ┌─────────────────────────┐
                        ╱ If ╲──────▶│ Set Bit-0 of Input by ORing│
                        ╲    ╱        │    Input with 0x01       │
                         ╲  ╱         └─────────────────────────┘
                          ╲╱
                           │
                           ▼
                    ┌─────────────────────────┐
                    │      Return Input        │
                    └─────────────────────────┘
                                │
                                ▼
                    ┌─────────────────────────┐
                    │   End Input Calculation  │
                    └─────────────────────────┘
```
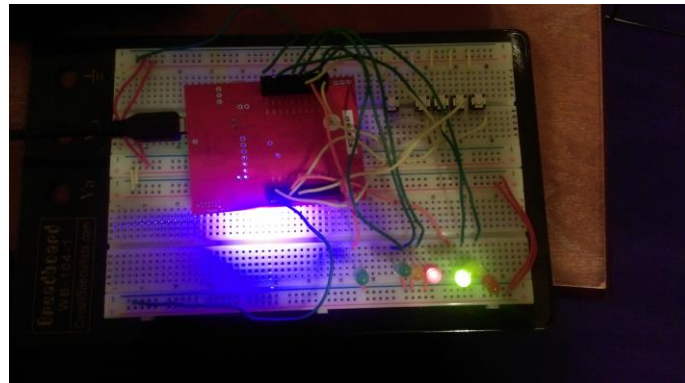
III.　Conclusion

In this experiment we learnt about Finite State Machine (FSM) and its implementation using C language. We also learnt about initialization and implementation SysTick and PLL.
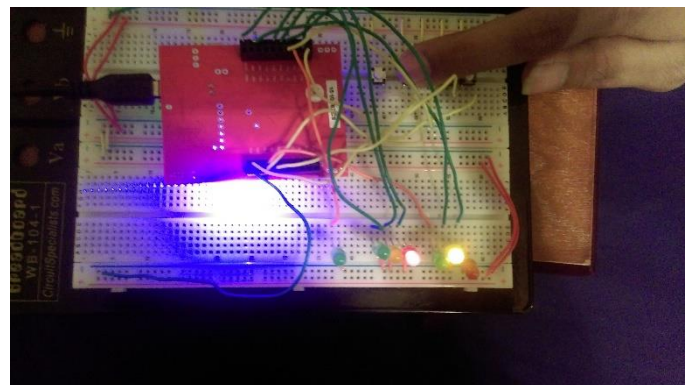
The System was Designed and Developed successfully.

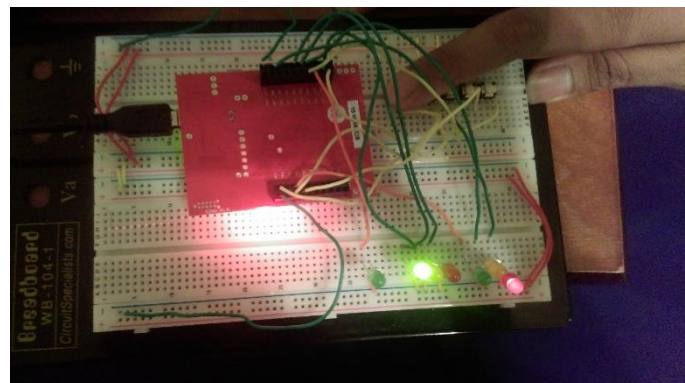Following are the Images of outputs during different states:
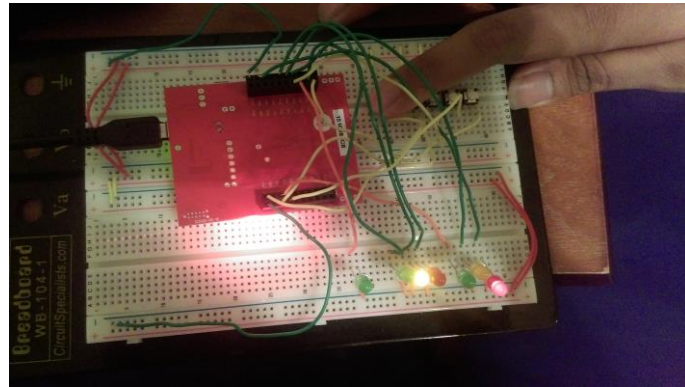
1. Go Side

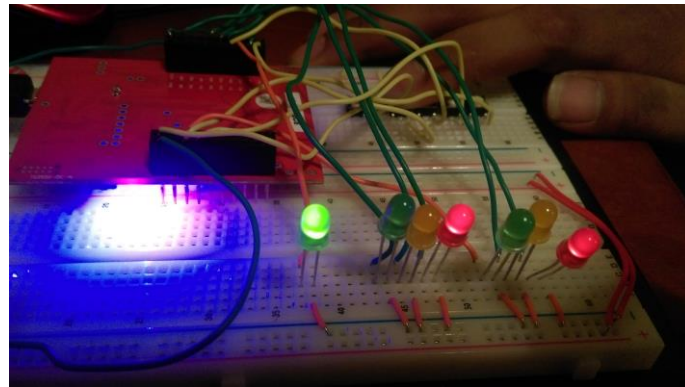
2. Wait Side

3. Go Main

4. Wait Main

5. Walk

IV. Reference

1. Figure#1:   F15LAB03 question provided.
2. PLL Configuration: Jonathan Valvano. Embedded Systems (Introduction to Arm\
                    xae Cortex\u2122-M Microcontrollers) (Kindle Locations
                    6577-6584). Jonathan Valvano.
3. Table#3:   Jonathan Valvano. Embedded Systems (Introduction to Arm\
                    xae Cortex\u2122-M Microcontrollers) (Kindle Locations
                    6727-6730). Jonathan Valvano.