Abu Dhabi University

Microprocessors and Firmware Programming

Lab Report 4
Traffic Light Crossing Signal Implementation

Author:
Muhammad Obaidullah
1030313

Supervisor:
Dr. Mohammed Assad Ghazal

Section 1

June 16, 2012

Abstract

In this Lab we solved a traffic signal problem, built hardware for the solution using ATMega16 micro-controller, coded the program for the solution in AVR Studio 4.19, and simulated the Solution in HAPSIM.

1 Introduction

In this Lab we had to Solve a traffic signal problem wherein we :-

1. Defined our Traffic light problem.
2. Constructed the finite state machine and the code flow diagram.
3. Wrote and compiled the C code for the problem.
4. Simulated the code using HAPSIM.
5. Downloaded our code to the ATMEGA 16.
6. Connected switches and LEDs to the ATMEGA 16.
7. Tested and checked if the LEDS correspond to the defined time in the ATMEGA 16 and the button inputs.

The Question was: Create a traffic light controller and test it using different types of LEDs and switches. There are two perpendicular streets to control taffic in. An East-West Street (EW) and a North-South Street (NS). Each street gets 30 seconds of green light. There are two push-buttons and one switch used in the system. The push-buttons are used by pedestrians wishing to cross the EW and NS streets, respectively. When pressed, if the amount of time remaining for green is greater than 5 seconds, the remaining time is shrunk to 5 seconds only before the change is made. Notice also that transition from red to green in any given street is through a warning yellow stage for 5 seconds, like with a normal traffic light. The switch is used for emergency and when ON, will cause the red lights to flash in the two streets to indicate a case of emergency until the switch is OFF, in which case normal operation is resumed.

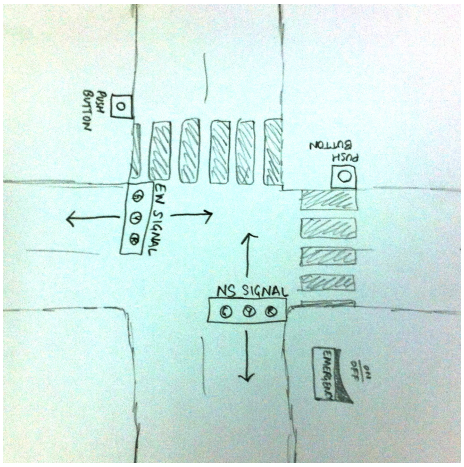

Figure 1: Understanding the situation problem

2 Experiment Set-up

The ATMega16 chip was already mounted on a safety bracket. We had to place the bracket with the micro-controller on to the breadboard. Then we connected the micro-processor to the JTAG MKII programmer,LEDs and the push-buttons as shown in the *Figure 1* and *Figure 2*.

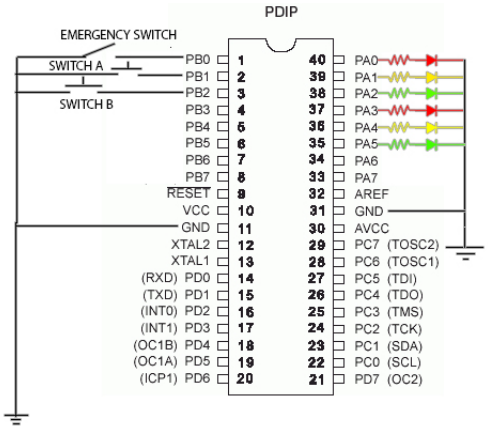

Figure 2: This is how we connect LEDs and the push buttons to the ATMega16

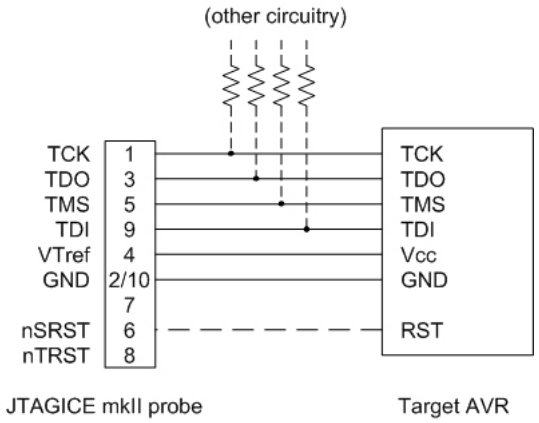

Figure 3: Figure showing connections from ATMega16 to the JTAG pins

3 List of Equipment used

- ATMega16 micro-controller chip.
- JTAG MKII programmer.
- Wires.
- Breadboard.
- Mounting bracket for micro-controller.
- 2 Green LEDs.
- 2 Yellow LEDs.
- 2 Red LEDs.
- 6 330 Ω resistors.
- 2 push buttons.
- 1 switch buttons.
- 5V power supply.
- AVR Studio IDE.
- HAPSIM.

4 Procedure

4.1 Making the FSM

Table 1: FSM Inputs, Outputs, and States

States	Inputs	Outputs
Emergency	EW Push Button	EW Red LED
East WestMoving	NS Push Button	EW Yellow LED
East West Warning	Emergency Switch	EW Green LED
North South Moving	Timer	NS Red LED
North South Warning		NS Yellow LED
		NS Green LED
		Timer

4.2 Converting the FSM to Code.

- Start AVR Studio and click on File/New/New Project.
- Write the following code into the AVR .c file.

```
#define F_CPU 1000000UL /* 1 MHz Internal Oscillator */
#include<avr/io.h>
#include<util/delay.h>
#include<avr/interrupt.h>

enum{EW_Moving, EW_Warning, NS_Moving, NS_Warning, Emergency};

int Current = EW_Moving;
int Count = 0;


#define LEDPORT PORTA
#define LEDPORTCONFIG DDRA
#define SWITCHPORT PINB
#define SWITCHPORTCONFIG DDRB

ISR(INT0_vect)
{


        int x = 5;
        int y = 10;

        if(Count < 25)
                Count = 25;

}
void doStates()
```

```
{
        Count++;

        switch(Current)
        {

                        case EW_Moving:
                                // 0 0 NSG NSY NSR EWG EWY EWR
                                LEDPORT = 0b00001100;
                                if((SWITCHPORT & 0x01) == 0x01)
                                {Current = Emergency; PORTB=0x00;}
                                if((SWITCHPORT & 0x02) == 0x02) if (Count < 25) Count = 25;
                                if (Count > 30)
                                {
                                        Count = 0;
                                        Current = EW_Warning;
                                }
                        break;

                        case EW_Warning:
                                                        // 0 0 NSG NSY NSR EWG EWY EWR
                                LEDPORT = 0b00001010;
                                if((SWITCHPORT & 0x01) == 0x01)
                                {Current = Emergency; PORTB=0x00;}
                                if (Count > 5)
                                {
                                        Count = 0;
                                        Current = NS_Moving;
                                }

                        break;

                        case NS_Moving:
                                                        // 0 0 NSG NSY NSR EWG EWY EWR
                                LEDPORT = 0b00100001;
                                if((SWITCHPORT & 0x01) == 0x01)
                                {Current = Emergency; PORTB=0x00;}
                                if((SWITCHPORT & 0x04) == 0x04) if (Count < 25) Count = 25;
                                if (Count > 30)
                                {
                                        Count = 0;
                                        Current = NS_Warning;
                                }

                        break;

                        case NS_Warning:
                                                        // 0 0 NSG NSY NSR EWG EWY EWR
                                LEDPORT = 0b00010001;
                                if((SWITCHPORT & 0x01) == 0x01)
                                {Current = Emergency; PORTB=0x00;}
                                if (Count > 5)
                                {
                                        Count = 0;
```

```
                                        Current = EW_Moving;
                                }

                        break;

                        case Emergency:

                                LEDPORT = LEDPORT ^ 0x09;
                                if((SWITCHPORT & 0x01) == 0x00)
                                {Current = EW_Moving; PORTB=0x00;}

                        break;


        }


}


int main()
{
        LEDPORTCONFIG = 0xFF;


        GICR = 0x40;
        MCUCR= 0x02;

        sei();

        while(1)
        {
                _delay_ms(1000);
                doStates();
        }


}
```

4.3 Simulating the Code.

- Open HAPSIM.
- Choose your Micro-controller, in this case its ATMega16.
- Open 6 LEDs and set them to Port A0,A1,A2,A3,A4,A5,A6, and A7.
- Open 3 buttons and set them to Pin B0,B1, and B3.
- Simulate your code by opening the .c file you saved from AVR studio.

4.4 Uploading the code to ATMega16.

- Connect JTAG to the computer through a USB cable and connect the JTAG Pins to the micro-controller.

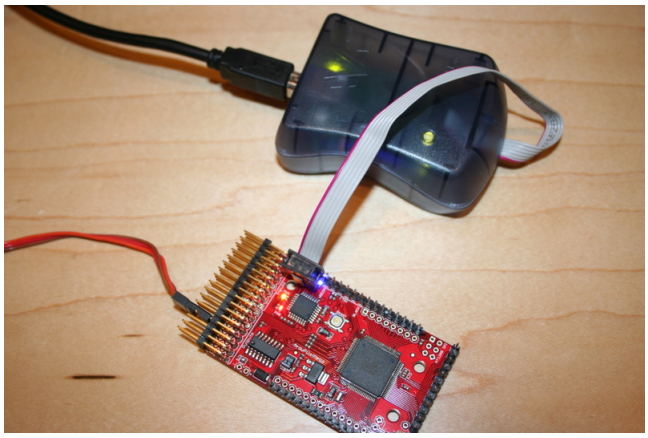

Figure 4: Connecting JTAG MKII to the ATMega 16

- connect the LEDs and the push-buttons to the ATMega16 as shown in *fiqure 2*.
- Click build and compile in AVR Studio.
- Run the code.

5 Results and Discussions

At the end of these exercises we got the following results:-

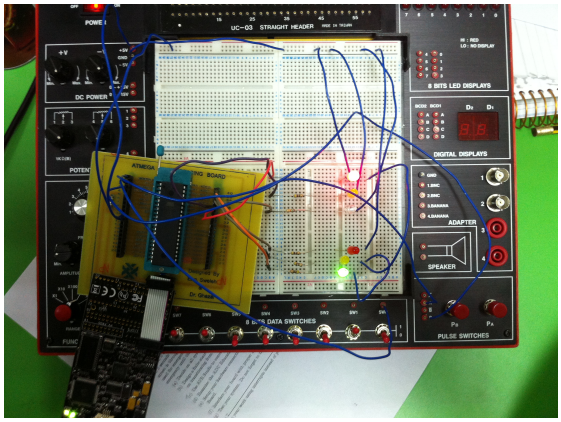

Figure 5: East-West is red while North-South is green.

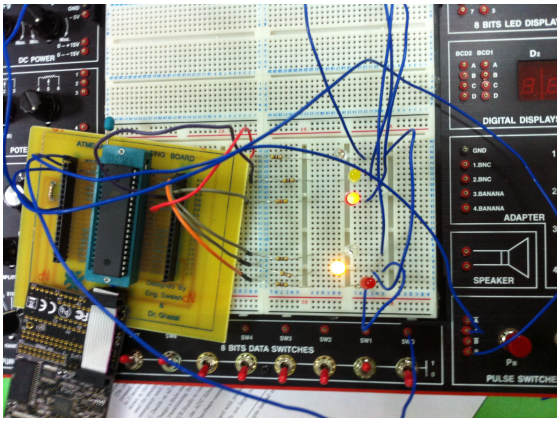

Figure 6: North-South is in warning state after being Green for 30 seconds

- Successful operation of a traffic signal was achieved.
- If we use interrupts, that keeps the micro-controller free for doing other useful tasks rather than keeping it busy in reading the inputs hundred times per second.

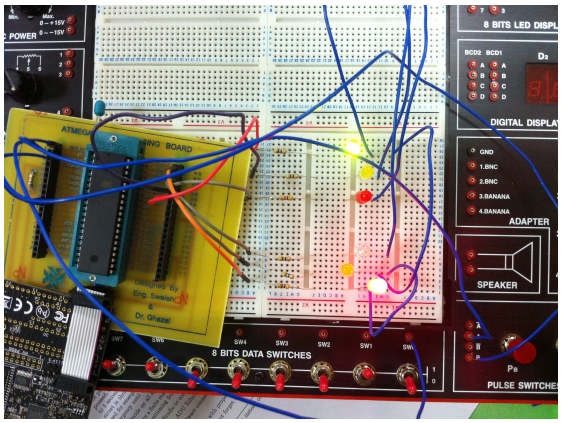

Figure 7: East-West is green while North-South is Red.

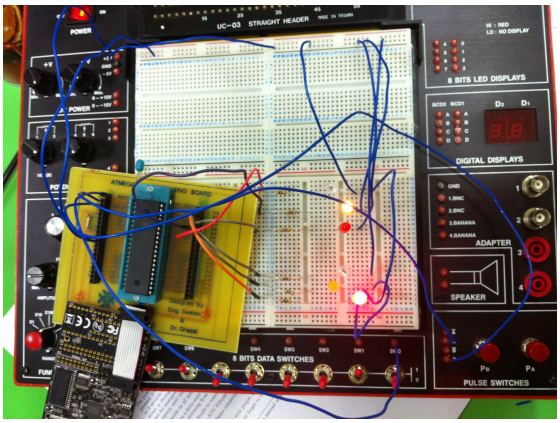

Figure 8: East-West is in warning state after being Green for 30 seconds

- For enabling interrupts we have to call a function sei()
- For using interrupts we have to include a header file "avr/interrupt.h".
- GICR stands for "General Interrupt Control Register" which filters the interrupts micro-controller should be listening too.
- Every Signal used to stay green for 30 seconds until or unless a push-button was pressed, in which case it would shorten its time to 5 seconds if the remaining time was greater than 5 seconds.
- In any point in time if emergency button was On (i.e. PORT B pin 0 was High) the red LEDs started toggling their state with the time duration of 1 sec because after every 1 second we are performing the dostates() function.

6 Conclusion

- All the LEDs have to be connected to the Port A because, we have set the Port A as output and not the other ports.
- Red, yellow, and green LEDs have to be connected to particular pins for the correct output .
- The LEDs have to be connected in series with a 330 Ω resistor in order to prevent LED from being burned down of high current coming from the micro-controller pins.
- The ATMEGA16/32 has 3 external interrupt lines : INTO,INT1 and INT2, on pins PD2,PD3 and PB2 therefore, only three external interrupts can be connected to the ATMega16 micro-controller.