EE443 - Embedded Systems

Experiment **1** Laboratory Report
Compilation and Simulation

Name: M. Serdar Karaman

Number: 190206038

Date: 02 - November - 2016

Objective

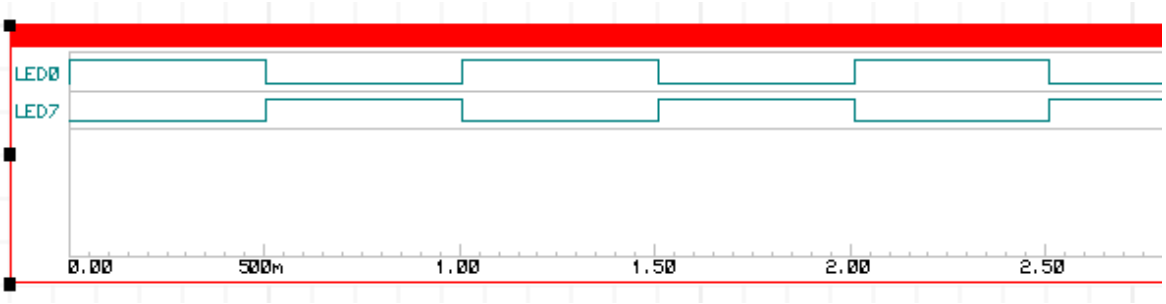
Writing codes on Code::Blocks and compiling codes using WinAVR compiler. Reading and understanding assembly code (.lss file).

Experimental Work

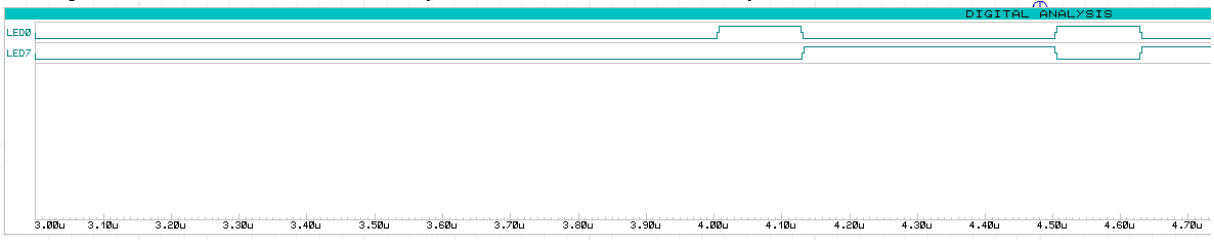
```
#include <avr/io.h>
#include <util/delay.h>

int main(void){
    DDRB = 0b11111111; // Tristate drivers for port B pins. 1 means output, 0
means input.
    while(1){
        PORTB = 0b00001111; //1-4 bits of Port B is high, others low.
        _delay_ms(500); //wait for .5 second
        PORTB = 0b11110000; //5-8 bits of Port B is high, others low.
        _delay_ms(500); //wait for .5 second
    };
    return 0;
}
```

I created a new project with required settings (8MHz clock frequency …). I compiled blink code with 500ms delay. Later I added Hex code to Protheus design. Simulation graph is below. Period is 1sec, nothing is unexpected.



Delays are deleted, code compiled and simulation repeated. Simulation is below.



It takes 4u second to start blinking. And period of signal is 0.5u second. (2MHz=¼ of clock frequency).

.lss is below.

```
00000080 <main>:
  80:8f ef          ldi  r24, 0xFF  ; 255
  82:84 b9          out  0x04, r24  ; 4
  84:9f e0          ldi  r25, 0x0F  ; 15
  86:80 ef          ldi  r24, 0xF0  ; 240
  88:95 b9          out  0x05, r25  ; 5
  8a:85 b9          out  0x05, r24  ; 5
  8c:fd cf          rjmp .-6        ; 0x88 <main+0x8>
```

1- Load 0xFF data to register24
2- Store register24 value to 0x04(PortB tristate driver)
3- Load 0x0F data to register25
4- Load 0xF0 data to register24 (register value is overwritten)
5- Store register25 value to 0x05(Port B Data Register) (1 clock)
6- Store register24 value to 0x05(Port B Data Register) (1 clock)
7- Relative Jump to 6 location back (to 0x88) (2 clock)

Results and Conclusion

Maximum frequency for blinking led is ¼ of clock frequency. But in this case high and low durations are not equal because rjmp operation takes 2 clock. So at the end result is 1 clock high and 3 clock low. To make this is equal we need to add 2 clock operation before low command. In this way max frequency will be 1/6 of clock frequency.

Answers to the Questions

Q1- 4 clock cycle. 2 for loading high and low values. 2 for jumping back. Period is 0.5u sec. It is 2Mhz like expected.
Q2- We need to extra wait 2 clock before low command. I added 2 more "**`PORTB = 0b00001111;`**".
Q3- No. I checked all load and jump operations on datasheet. LDI takes 1 clock. LD operations takes 2 clock. RJMP takes 2 clock. I tough maybe I can use direct jump but it takes 3 clock. So this code is best option in our case.