INSTITUTE OF ENGINNERING , CENTRAL CAMPUS,PULCHOWK

EMBEDDED SYSTEM

LAB #3

Programming Timers of 8051/8052 Microcontroller

Submitted BY:
Amrit Prasad Phuyal
Roll: PULL074BEX004

Submitted To:
Department of Electronics and
Computer Engineering

November 13, 2020

Contents

List of Figures

List of Codes

1 Introduction

1.1 Microcontroller

A microcontroller is an integrated circuit (IC), usually via an MPU, memory and certain peripherals, to control other parts of an electronic system . These devices are optimized for embed-in applications that require agile and agile processing, digital, analog or electromechanical interactions.

1.2 8051 Microcontroller

In 1981, Intel introduced an 8-bit microcontroller called the 8051. It was referred as system on a chip because it had 128 bytes of RAM, 4K byte of on-chip ROM, two timers, one serial port, and 4 ports (8-bit wide), all on a single chip.

The different features of the 8051 microcontroller include:

- 4KB bytes on-chip program memory (ROM)
- 128 bytes on-chip data memory (RAM)
- Four register banks
- 128 user defined software flags
- 8-bit bidirectional data bus
- 16-bit unidirectional address bus
- 32 general purpose registers each of 8-bit
- 16 bit Timers (usually 2, but may have more or less)
- Three internal and two external Interrupts
- Four 8-bit ports,(short model have two 8-bit ports)
- 16-bit program counter and data pointer
- 8051 may also have a number of special features such as UARTs, ADC, Op-amp, etc.

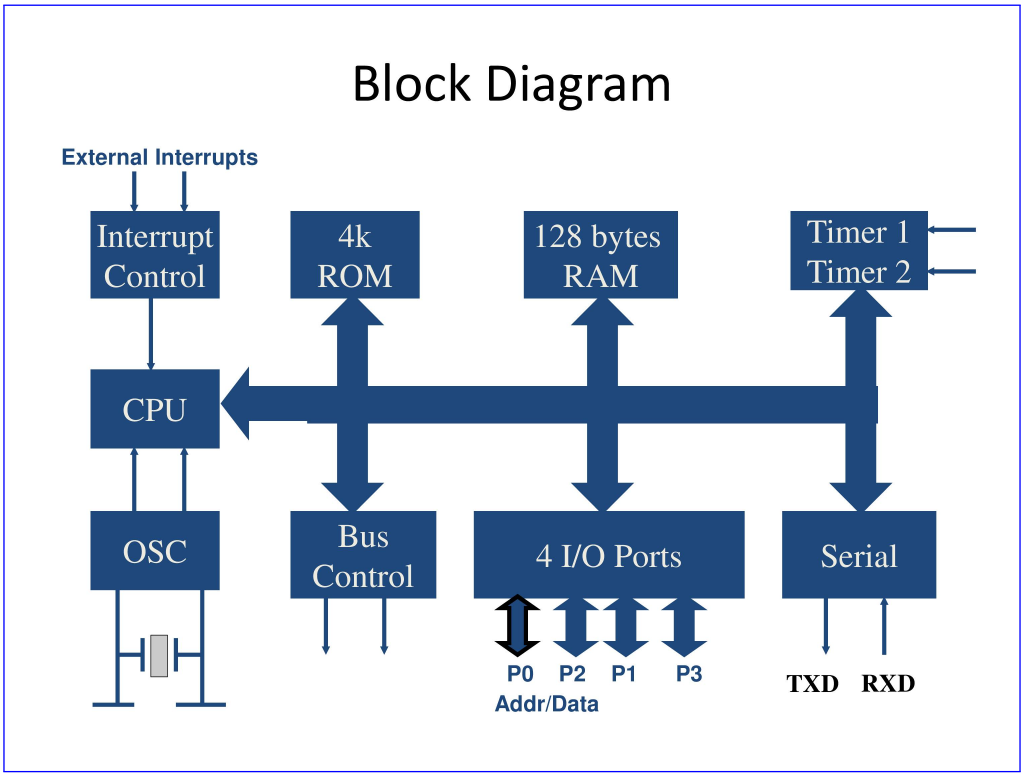

Figure 1: Block diagram of 8051 microcontroller

1.3 7-Segment LED Display

A seven segment display module is an electronic device used to display digital numbers and it is made up of seven LED segments. LEDs are PN-junction diodes which emit energy by a process called electroluminescence. Because of the small size of the LEDs, it is really easy for a number of them to be connected together to make a unit like seven segment display. The light energy is emitted as 'photons' when it is forward biased by a voltage applied across its junctions. In a seven segment display module, seven LED s are arranged in a rectangle. Sometimes, an additional LED is seen in a seven segment display unit which is meant for displaying a decimal point.

Features of seven segment Display:-

- Available in two modes Common Cathode (CC) and Common Anode (CA)
- Available in many different sizes like 9.14mm,14.20mm,20.40mm,38.10mm,57.0mm and 100mm (Commonly used/available size is 14.20mm)
- Available colours: White, Blue, Red, Yellow and Green (Res is commonly used)
- Low current operation
- Better, brighter and larger display than conventional LCD displays.
- Current consumption : 30mA / segment
- Peak current : 70mA

The displays common pin is generally used to identify which type of 7-segment display it is. As each LED has two connecting pins, one called the "Anode" and the other called the "Cathode", there are therefore two types of LED 7-segment display called: Common Cathode (CC) and Common Anode (CA). The difference between the two displays, as their name suggests, is that the common cathode has all the cathodes of the 7-segments connected directly together and the common anode has all the anodes of the 7-segments connected together

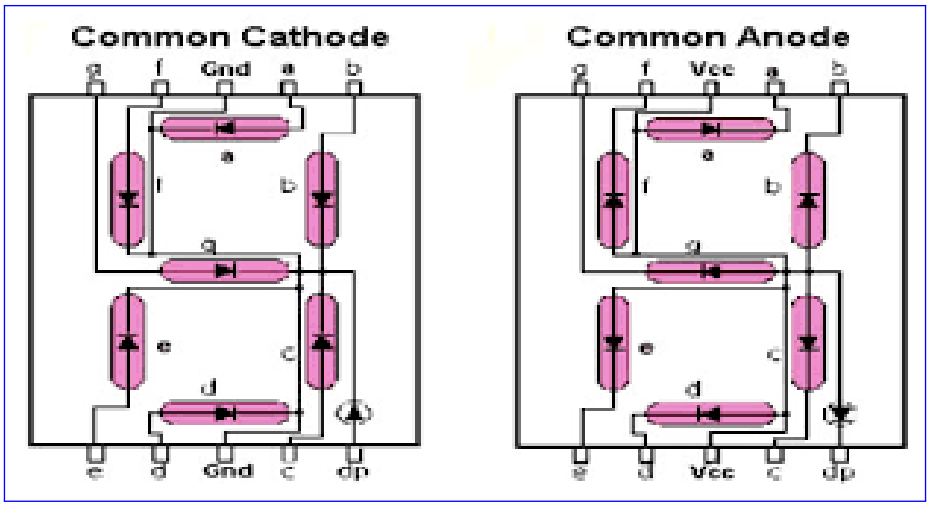

Figure 2: Common cathode vs Common anode 7 segment display

Numbers	Common Cathode		Common Anode	
	(DP)GFEDCBA	HEX Code	(DP)GFEDCBA	HEX Code
0	00111111	0x3F	11000000	0xC0
1	00000110	0x06	11111001	0xF9
2	01011011	0x5B	10100100	0xA4
3	01001111	0x4F	10110000	0xB0
4	01100110	0x66	10011001	0x99
5	01101101	0x6D	10010010	0x92
6	011111101	0x7D	10000010	0x82
7	00000111	0x07	11111000	0xF8
8	01111111	0x7F	10000000	0x80
9	01101111	0x6F	10010000	0x90

Figure 3: Lookup table for Common anode and Common Cathode

1.4 Applications of Seven Segment Display

- Used in applications where font size is required to be bigger
- Microcontroller Independent, hence used in small circuit projects
- Used in combination with four segments to display measurement/sensor value with four characters
- Has bright illumination, hence used where display are required to work in low light or dark conditions

1.5 Timers in 8051

The basic 8051 has two on-chip timers that can be used for timing duration or for counting external events. Interval timing allows the programmer to perform operations at specific instants in time. Since the microcontroller operates at a specific frequency, we could work out exactly how much iterations of the time delay was needed to give us the required delay. Their application could be in communication for generating rectangle pulses, watchdog timer, in manufacturing industry for counting objects, measuring intervals, etc. There are two different types of timer: Interval timer and Counter. 5

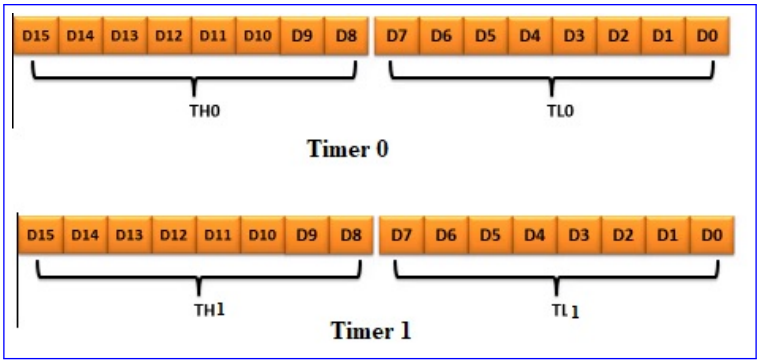

Figure 4: T0, T1 Timer Register

Two timers, namely Timer 0 and Timer 1 are 16 bits timer and since 8051 has an 8-bit architecture, each 16-bits timer is accessed as two separate registers of low byte and high byte. The low byte register is called TL0/TL1 and the high byte register is called TH0/TH1.

1.6 Timer Mode Register(TMOD)

TMOD is a 8-bit register whose lower 4 bits are for Timer 0 and upper 4 bits are for Timer 1. It is byte addressable only, which is loaded at the very beginning of a program to initialize a timer's mode. In each case, the lower 2 bits are used to set the timer mode and upper 2 bits to specify the operation.

- **Timer Mode 0** Mode 0is identical for Timer 0 and Timer 1. Both timers work as 13-bit counters;an interrupt is generated when counter overflows. It takes 8192 input pulses to generate the next interrupt. Timers use 8-bits of THi and 5 lower bits of TLi. After timer overflows TFi(Timer flag in TCON) is set, hence an interrupt occurs.
- **Timer Mode 1** This mode is similar to mode 0. This timer uses all 8 bits of THi and 8 bits of TLi. So it is a 16-bit counter which can take 65536 input pulses to generate the next interrupt.
- **Timer Mode 2** In this mode, the timers are 8 bits auto reload type. The timer is operated by TLi, when TLi overflows again it is automatically loaded by THi. So the initial value is loaded to the THi register at first.
- **Timer Mode 3** In this mode, only timer 0 can be used. This is also called split timer mode. Timer 0 operates TL0 and TH0 as two separate 8 bit timers/counters. Timer 0with TL0 is operated with TF0 and TR0 while timer 0 with TH0 is operated with TF1and TR1.

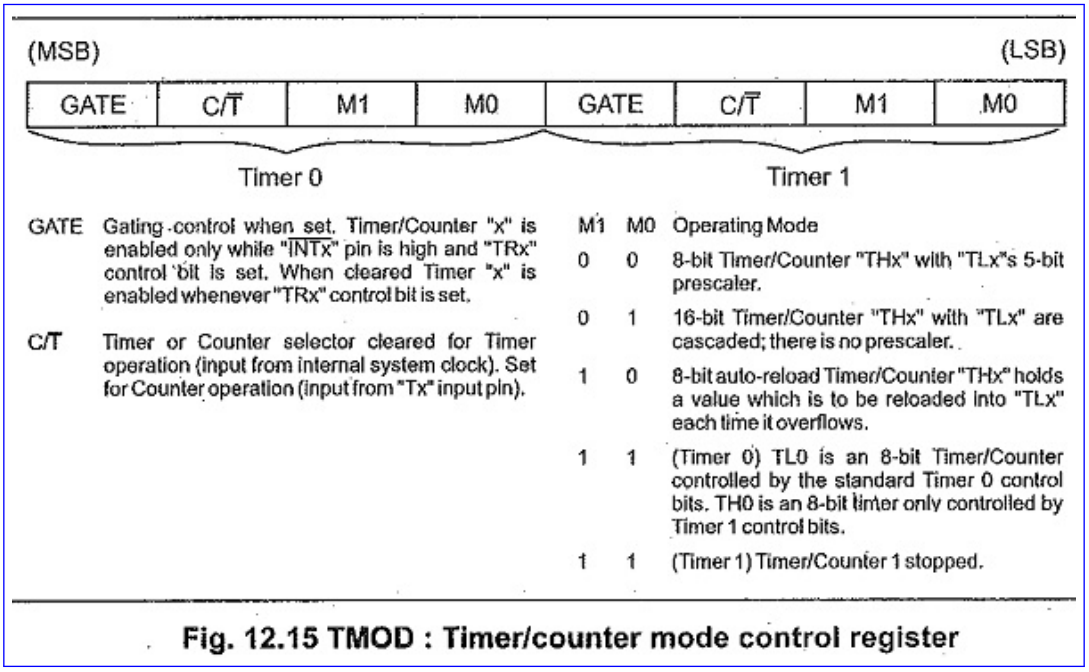

Figure 5: Timer mode control register

1.7 Timer Control Register (TCON)

The 8051 microcontroller has one 8-bit register that holds the timer flags, interrupt ags and timer run control bit. This register is bit addressable and is used by both timers as well as interrupts. The timers use the upper4-bits while interrupts use the lower 4-bits.

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 Overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn timer/counter on/off.
TF0	TCON.5	Timer 0 Overflow Flag. Set by hardware on timer/counter overflow. Cleared when interrupt processed.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn timer/counter on/off.
IE1	TCON.3	Interrupt 1 Edge Flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge Flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.

Figure 6: Timer Control Register (TCON)

1.8 Clock Sources for Timer

Using TMOD register, timer operation is selected, and timer is clocked from an oscillator. Frequency for timer is $1/12^{th}$ the frequency of the crystal attached to the 8051 microcontroller, which is equivalent to 921.6 KHz (frequency of an oscillator is 11.0592 MHz). This is so as in8051 microcontroller, 12 oscillator periods constitute a machine cycle. Hence machine cycle period is 1.085 microseconds.

2 Objectives

To enable us to write assembly language code for the 8051/8052 micro-controller capableof:

- Applying timers in different timing modes.
- Implementing accurate delays using timers.

3 Equipment Required

- Hardware: 8051 or 8052 micro-controller development board, Jumper cables
- Simulation Software: KEIL, Vision-Embedded development tool, Proteus Design Suite –Professional PCB layout, circuit design and simulation tool
- In-System Programming (ISP) Software: ProgISP –An in-system-programmable tool to load HEX files in to micro-controller
- Device Drivers: LibUSB –Application controlling data transfer to/from USB devices

4 LAB problems

4.1 Question -1

Generate a periodic square wave having a period of 15 ms and a duty cycle of 20 % . The waveform should be produced at pin zero of port two (P2.0). The XTAL frequency is 11.0592 MHz. Observe the waveform on an oscilloscope and measure the ON and OFF timers.

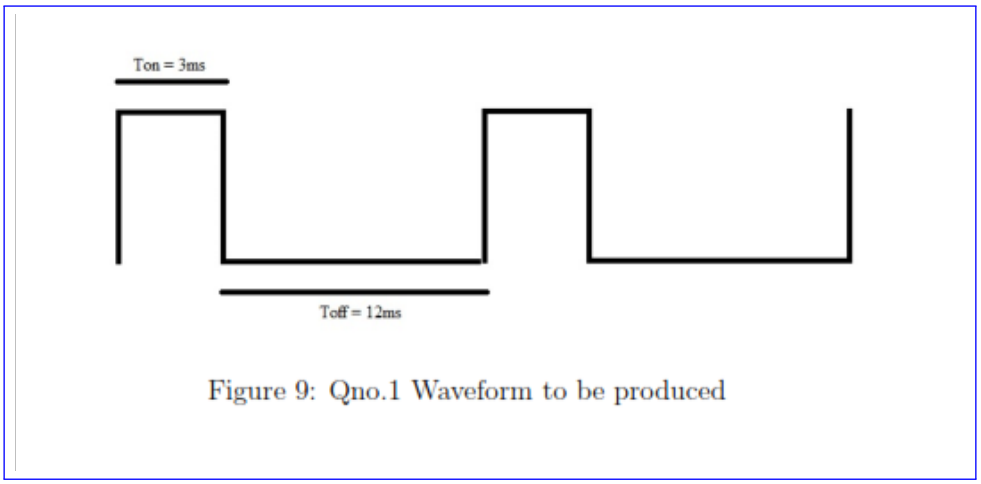

Figure 9: Qno.1 Waveform to be produced

Figure 7: Waveform to be generated for Problem 1

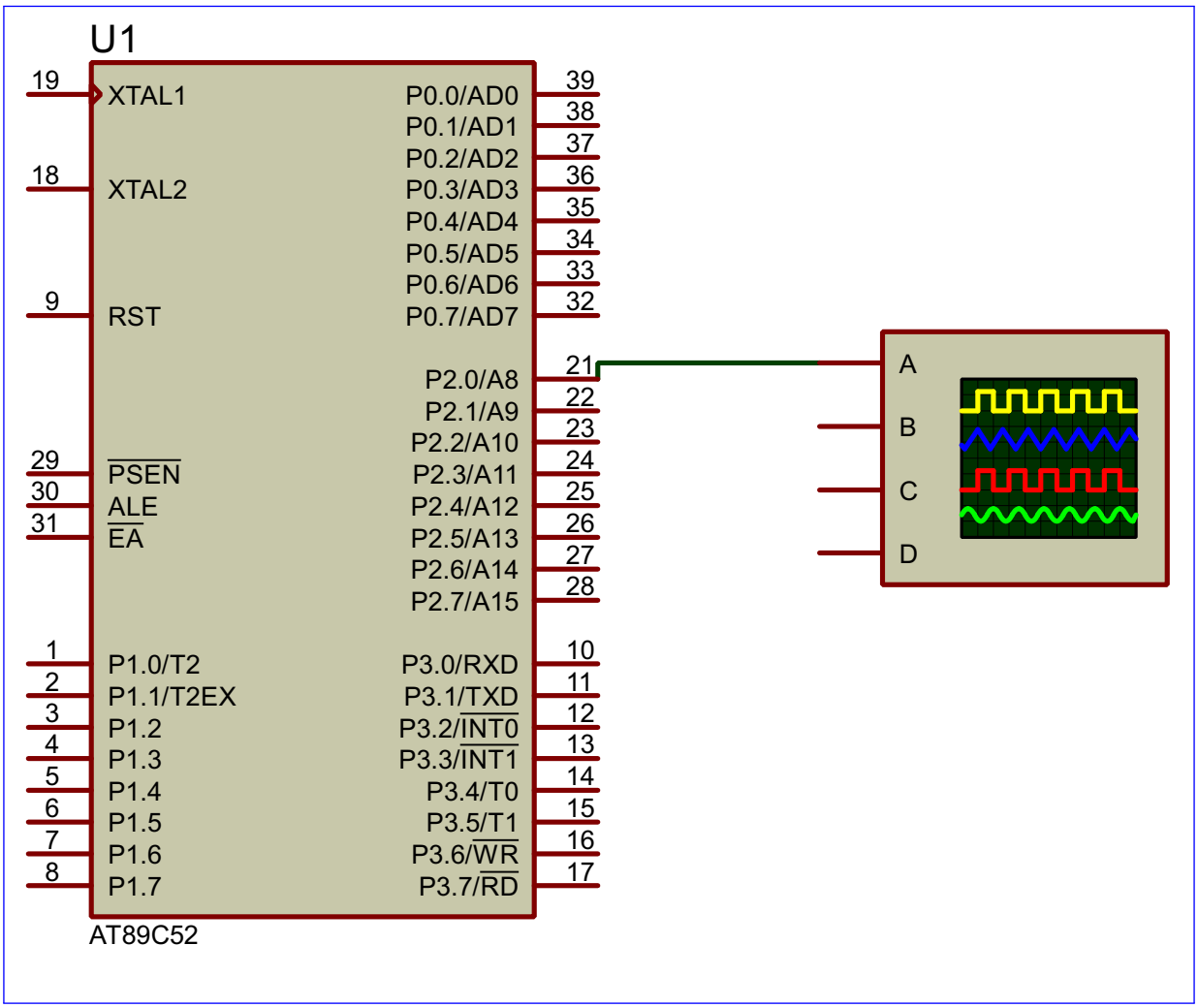

Figure 8: Proteus Schematic Problem no.1

1. **Using Timer 1 in mode 0 (13-bit timer mode)**

Assembly

```
ORG 00H
MOV TMOD,#00
HERE: MOV TL1,#14H
MOV TH1,#0A9H
SETB P2.0
ACALL DELAY
MOV R2,#04H
AGN: MOV TL1,#14H
MOV TH1,#0A9H
CLR P2.0
ACALL DELAY
DJNZ R2,AGN
SJMP HERE
DELAY: SETB TR1
AGAIN: JNB TF1, AGAIN
CLR TR1
CLR TF1
RET
END
```

Code 1: Problem no. 1.a Assembly

C language

```
#include <reg51.h>
sbit select_bit = P2 ^ 0;
void delay(void)
{
    TMOD = 0x00;
    TL1 = 0x14;
    TH1 = 0xA9;
    TR1 = 1;
    while (!TF1)
        ;
    TR1 = 0;
    TF1 = 0;
}
void main(void)
{
    int i;
    while (1)
    {
        select_bit = 1;
        delay();
        for (i = 0; i < 4; i++)
        {
            select_bit = 0;
            delay();
        }
    };
}
```

Code 2: Problem no. 1.a C language

2. **Using Timer 0 is mode 1 (16-bit timer mode)**

Assembly

```
ORG 00H
MOV TMOD,#01
HERE: MOV TL0,#34H
MOV TH0,#0F5H
SETB P2.0
ACALL DELAY
MOV R2,#04H
AGN: MOV TL0,#34H
MOV TH0,#0F5H
CLR P2.0
ACALL DELAY
DJNZ R2,AGN
SJMP HERE
DELAY: SETB TR0
AGAIN: JNB TF0, AGAIN
CLR TR0 ;1
CLR TF0 ;1
RET ;2
END
```

Code 3: Problem no. 1.b Assembly

C language

```
#include <reg51.h>
sbit select_bit = P2 ^ 0;
void delay(void)
{
    TMOD = 0x01;
    TL0 = 0x34;
    TH0 = 0xF5;
    TR0 = 1;
    while (!TF0)
        ;
    TR0 = 0;
    TF0 = 0;
```

```
}
void main(void)
{
    int i;
    while (1)
    {
        select_bit = 1;
        delay();
        for (i = 0; i < 4; i++)
        {
            select_bit = 0;
            delay();
        }
    };
}
```

Code 4: Problem no. 1.b C language

3. **Using Timer 1 in mode 2 (8-bit auto-reload timer mode)**

Assembly

```
ORG 00H
MOV TMOD,#20H
HERE: MOV R2,#0FH
AGN: MOV TH1,#48H
SETB P2.0
ACALL DELAY
DJNZ R2,AGN
MOV R2,#3CH
AGN1: MOV TH1,#48H
CLR P2.0
ACALL DELAY
DJNZ R2,AGN1
SJMP HERE
DELAY: SETB TR1
AGAIN: JNB TF1, AGAIN
CLR TR1
CLR TF1
RET
END
```

Code 5: Problem no. 1.c Assembly

C language

```
#include <reg51.h>
sbit select_bit = P2 ^ 0;
void delay(void)
{
    int i;
    TMOD = 0x20;
    for (i = 0; i < 15; i++)
    {
        TH1 = 0x48;
        TR1 = 1;
        while (!TF1)
            ;
        TR1 = 0;
        TF1 = 0;
    }
}
void main(void)
{
    int i;
    while (1)
    {
        select_bit = 1;
        delay();
        for (i = 0; i < 4; i++)
        {
            select_bit = 0;
            delay();
        }
    };
}
```

Code 6: Problem no. 1.c C language

4. **Using Timer 0 (TL0) in mode 3 (8-bit split timer mode)**

Assembly

```
ORG 00H
MOV TMOD,#03H
HERE: MOV R2,#0FH
AGN: MOV TL0,#48H
SETB P2.0
ACALL DELAY
DJNZ R2,AGN
MOV R2,#3CH
AGN1: MOV TL0,#48H
CLR P2.0
ACALL DELAY
DJNZ R2,AGN1
SJMP HERE
DELAY: SETB TR0
AGAIN: JNB TF0, AGAIN
CLR TR0
CLR TF0
RET
END
```

Code 7: Problem no. 1.d Assembly

C language

```c
#include <reg51.h>
sbit select_bit = P2 ^ 0;
void delay(void)
{
    int i;
    TMOD = 0x03;
    for (i = 0; i < 15; i++)
    {
        TL0 = 0x48;
        TR0 = 1;
        while (!TF0);
        TR0 = 0;
        TF0 = 0;
    }
}
void main(void)
{
    int i;
    while (1)
    {
        select_bit = 1;
        delay();
        for (i = 0; i < 4; i++)
        {
            select_bit = 0;
            delay();
        }
    };
}
```

Code 8: Problem no. 1.d C language

OUTPUT: All Outputs are Screenshot of Proteus Simulation. Observation for mode 0, mode 1, mode 2 and mode 3 are below . Due to some error perfect measurement cannot be taken . we have to generate a periodic square wave having a period of 15 ms and duty cycle of 20 %. So, total on time of square wave per period T_{ON} is 3ms while T_{OFF} is 12 ms we have made a delay of 3 ms and used it for both on and off cycle. For off cycle, delay is looped four times as off time is four times of on time.

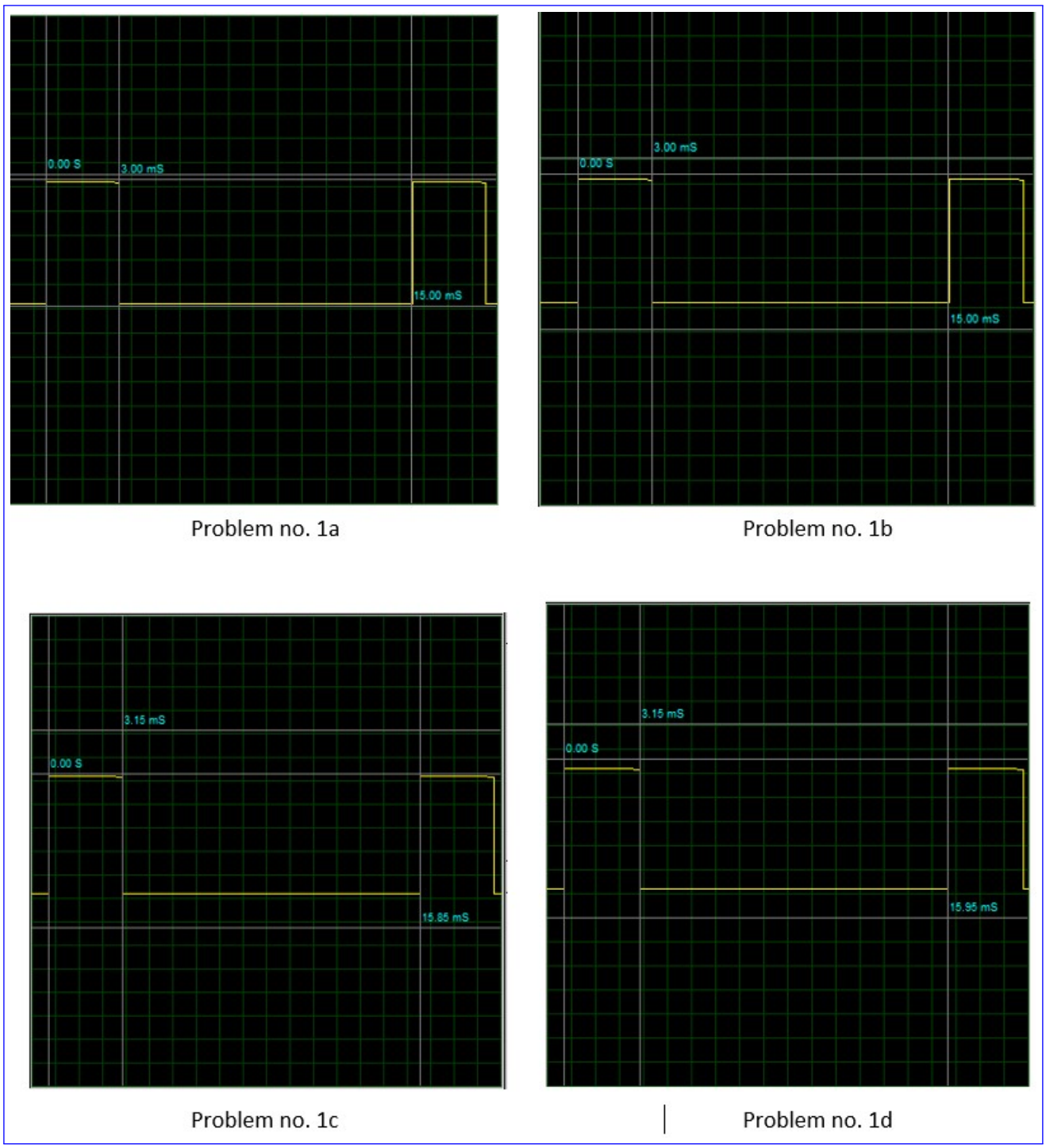

Figure 9: Graphs Problem no.1

4.2 Question -2

Generate the periodic waveform as shown in figure 11. The waveform should be produced at pin zero of port zero (P0.0). The XTAL frequency is 11.0592 MHz. Observe the waveform on an oscilloscope and measure the ON and OFF times.

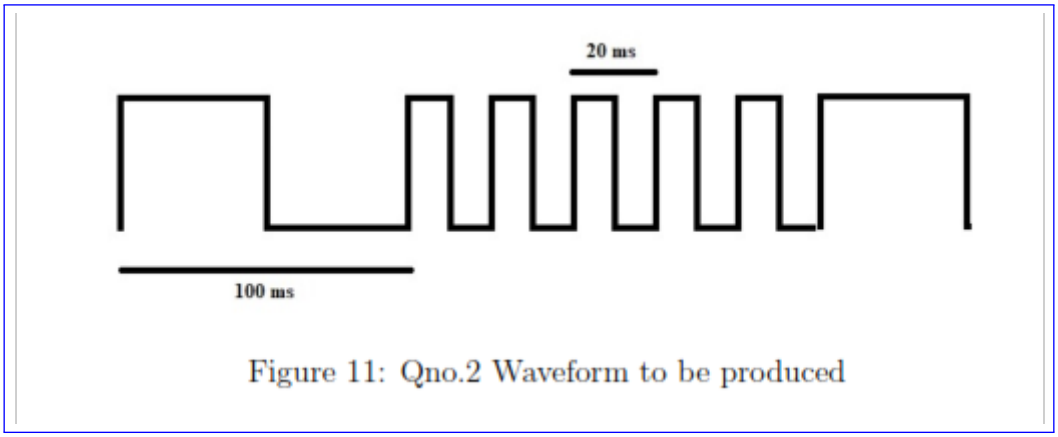

Figure 10: Waveform to be generated for Problem 2

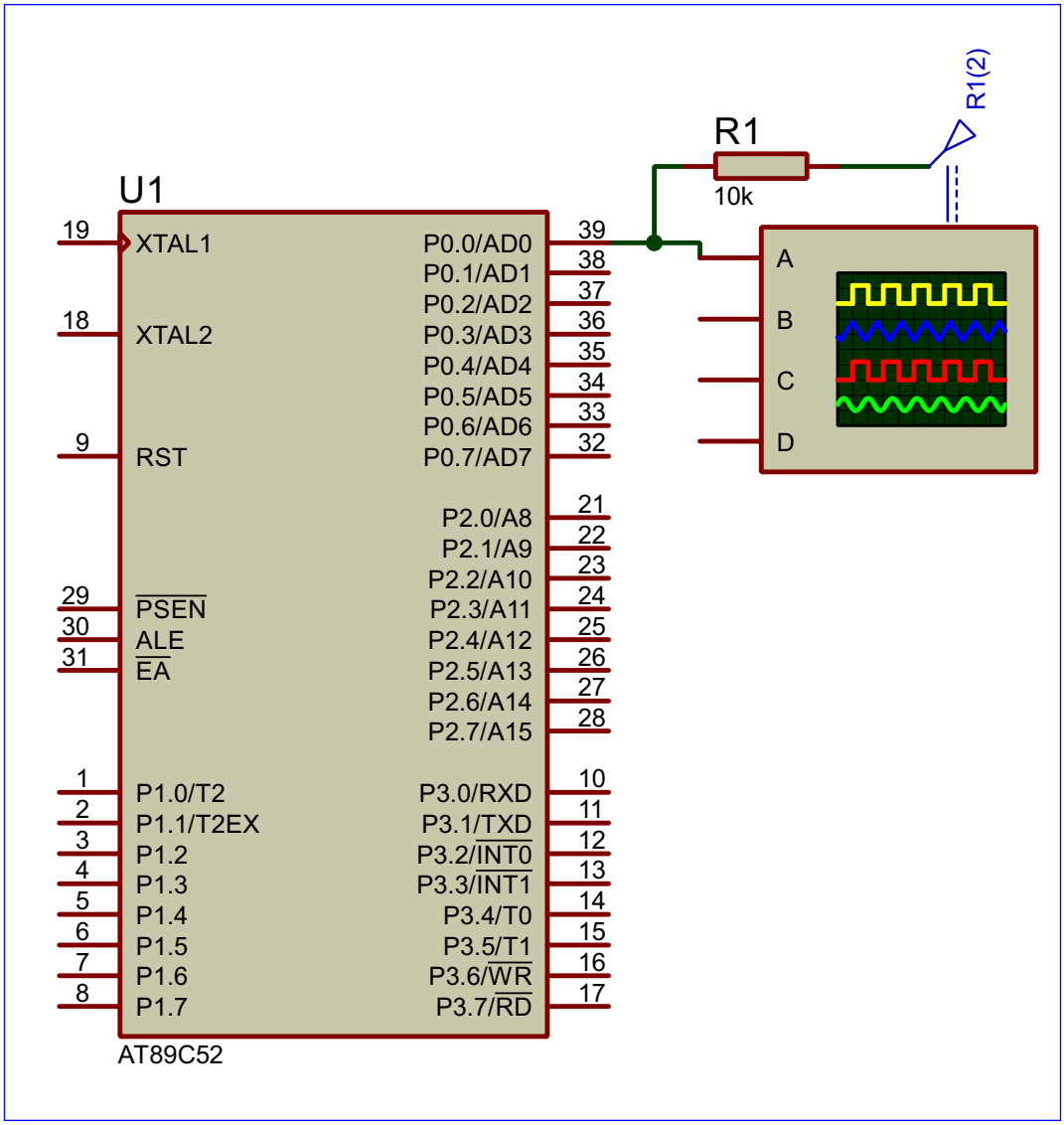

Figure 11: Proteus Schematic Problem no.2

1. **Using Timer 0 and mode 0 (13-bit timer mode)**

Assembly

```
ORG 00H
MOV TMOD,#00
REPEAT: MOV R2,#02H
LOOP1: MOV R1,#0AH
HERE1: MOV TL0,#00H
MOV TH0,#70H
ACALL DELAY
DJNZ R1,HERE1
CPL P0.0
DJNZ R2,LOOP1
MOV R2,#0AH
LOOP2: MOV R1,#02H
HERE2: MOV TL0,#00H
MOV TH0,#70H
ACALL DELAY
DJNZ R1,HERE2
CPL P0.0
DJNZ R2,LOOP2
SJMP REPEAT
DELAY: SETB TR0
AGAIN: JNB TF0, AGAIN
CLR TR0
CLR TF0
RET
END
```

Code 9: Problem no. 2.a Assembly

C language

```
#include <reg51.h>
sbit select_bit = P0 ^ 0;
void delay(int factor)
{
    int i;
    TMOD = 0x00;
    for (i = 0; i < factor; i++)
    {
        TL0 = 0x00;
        TH0 = 0x70;
        TR0 = 1;
        while (!TF0)
            ;
        TR0 = 0;
        TF0 = 0;
    }
}
void main(void)
{
    int i;
    while (1)
    {
        select_bit = 1;
        delay(10);
        select_bit = 0;
        delay(10);
        for (i = 0; i < 5; i++)
        {
            select_bit = 1;
            delay(2);
            select_bit = 0;
            delay(2);
        }
    };
}
```

Code 10: Problem no. 2.a C language

2. **Using Timer 1 in mode 1 (16-bit timer mode)**

Assembly

```
ORG 00H
MOV TMOD,#10H
REPEAT: MOV R2,#02H
HERE1: MOV TL1,#0FEH
MOV TH1,#4BH
ACALL DELAY
CPL P0.0
DJNZ R2,HERE1
MOV R2,#0AH
HERE2: MOV TL1,#00H
MOV TH1,#0DCH
ACALL DELAY
CPL P0.0
DJNZ R2,HERE2
SJMP REPEAT
DELAY: SETB TR1
AGAIN: JNB TF1, AGAIN
CLR TR1
CLR TF1
RET
END
```

Code 11: Problem no. 2.b Assembly

C language

```c
#include <reg51.h>
sbit select_bit = P0 ^ 0;
void delay(char TH, char TL)
{
    TH1 = TH;
    TL1 = TL;
    TMOD = 0x10;
    TR1 = 1;
    while (!TF1)
        ;
    TR1 = 0;
    TF1 = 0;
}
void main(void)
{
    int i;
    while (1)
    {
        select_bit = 1;
        delay(0x4B, 0xFE);
        select_bit = 0;
        delay(0x4B, 0xFE);
        for (i = 0; i < 5; i++)
        {
            select_bit = 1;
            delay(0xDC, 0x00);
            select_bit = 0;
            delay(0xDC, 0x00);
        }
    };
}
```

Code 12: Problem no. 2.b C language

3. **Using Timer 0 in mode 2 (8-bit auto-reload timer mode)**

Assembly

```
ORG 00H
MOV TMOD,#02H
LOOP: MOV R2,#02H
HERE1: MOV R1,#0C0H
HER1: MOV TH0,#1AH
ACALL DELAY
DJNZ R1,HER1
CPL P0.0
DJNZ R2,HERE1
MOV R2,#0AH
HERE2: MOV R1,#26H
HER2: MOV TH0,#1AH
ACALL DELAY
DJNZ R1,HER2
CPL P0.0
DJNZ R2,HERE2
SJMP LOOP
DELAY: SETB TR0
AGAIN: JNB TF0, AGAIN
CLR TR0
CLR TF0
RET
END
```

Code 13: Problem no. 2.c Assembly

C language

```c
#include <reg51.h>
sbit select_bit = P0 ^ 0;
void delay(int factor)
{
    int i;
    for (i = 0; i < factor; i++)
    {
        TMOD = 0x02;
        TH0 = 0x1A;
        TR0 = 1;
        while (!TF0)
            ;
        TR0 = 0;
        TF0 = 0;
    }
}
void main(void)
{
    int i;
    while (1)
    {
        select_bit = 1;
        delay(192);
        select_bit = 0;
        delay(192);
        for (i = 0; i < 5; i++)
        {
            select_bit = 1;
            delay(38);
            select_bit = 0;
            delay(38);
        }
    };
}
```

Code 14: Problem no. 2.c C language

4. **Using Timer 0 (TH0) in mode 3 (8-bit split timer mode)**

Assembly

```
ORG 00H
MOV TMOD,#03H
REPEAT: MOV R2,#02H
HERE1: MOV R1,#0C0H
HER1: MOV TH0,#1AH
ACALL DELAY
DJNZ R1,HER1
CPL P0.0
DJNZ R2,HERE1
MOV R2,#0AH
HERE2: MOV R1,#26H
HER2: MOV TH0,#1AH
ACALL DELAY
DJNZ R1,HER2
CPL P0.0
DJNZ R2,HERE2
SJMP REPEAT
DELAY: SETB TR1
AGAIN: JNB TF1, AGAIN
CLR TR1
CLR TF1
RET
END
```

Code 15: Problem no. 2.d Assembly

C language

```
#include <reg51.h>
sbit select_bit = P0 ^ 0;
void delay(int factor)
{
    int i;
    for (i = 0; i < factor; i++)
    {
        TMOD = 0x03;
        TH0 = 0x1A;
        TR1 = 1;
        while (!TF1)
            ;
        TR1 = 0;
        TF1 = 0;
    }
}
void main(void)
{
    int i;
    while (1)
    {
        select_bit = 1;
        delay(192);
        select_bit = 0;
        delay(192);
        for (i = 0; i < 5; i++)
        {
            select_bit = 1;
            delay(38);
            select_bit = 0;
            delay(38);
        }
    };
}
```

Code 16: Problem no. 2.d C language

OUTPUT: There are two waveforms that we have top produce. First waveform has on and off cycle of 50 ms while second waveform has on and off cycle of 10 ms. In above program, two separate delays are made for 50 ms and 10 ms delay by providing different values in timer register.Observation for mode 0, mode 1, mode 2 and mode 3 are below . Due to some error perfect measurement cannot be taken .

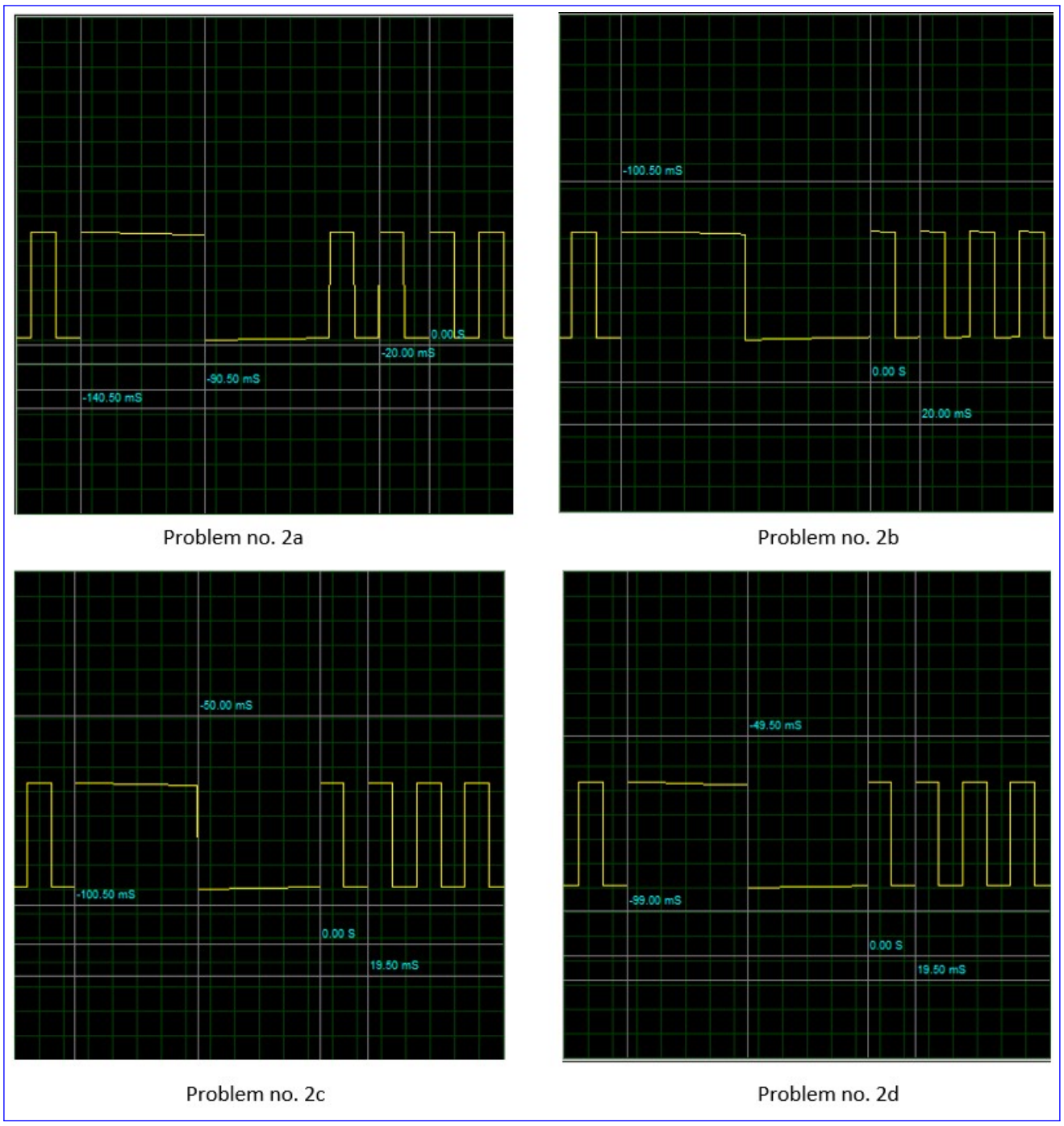

Figure 12: Graphs Problem no.2

4.3 Question -3

Design a digital minutes and seconds in double digit format. The clock should count from 00:00 to 59:59 and repeat. Time should be displayed in decimal format using four 7-segment LED units. A decimal point should separate minutes from seconds. Use an appropriate timer and timer mode. Use port 0 (P0) to send data to 7-segment LED units. Use transistors as switches to activate or deactivate the 7-segment LED units using pins 0, 1, 2 and 3 of port 2 (P2.0, P2.1, P2.2, P2.3).

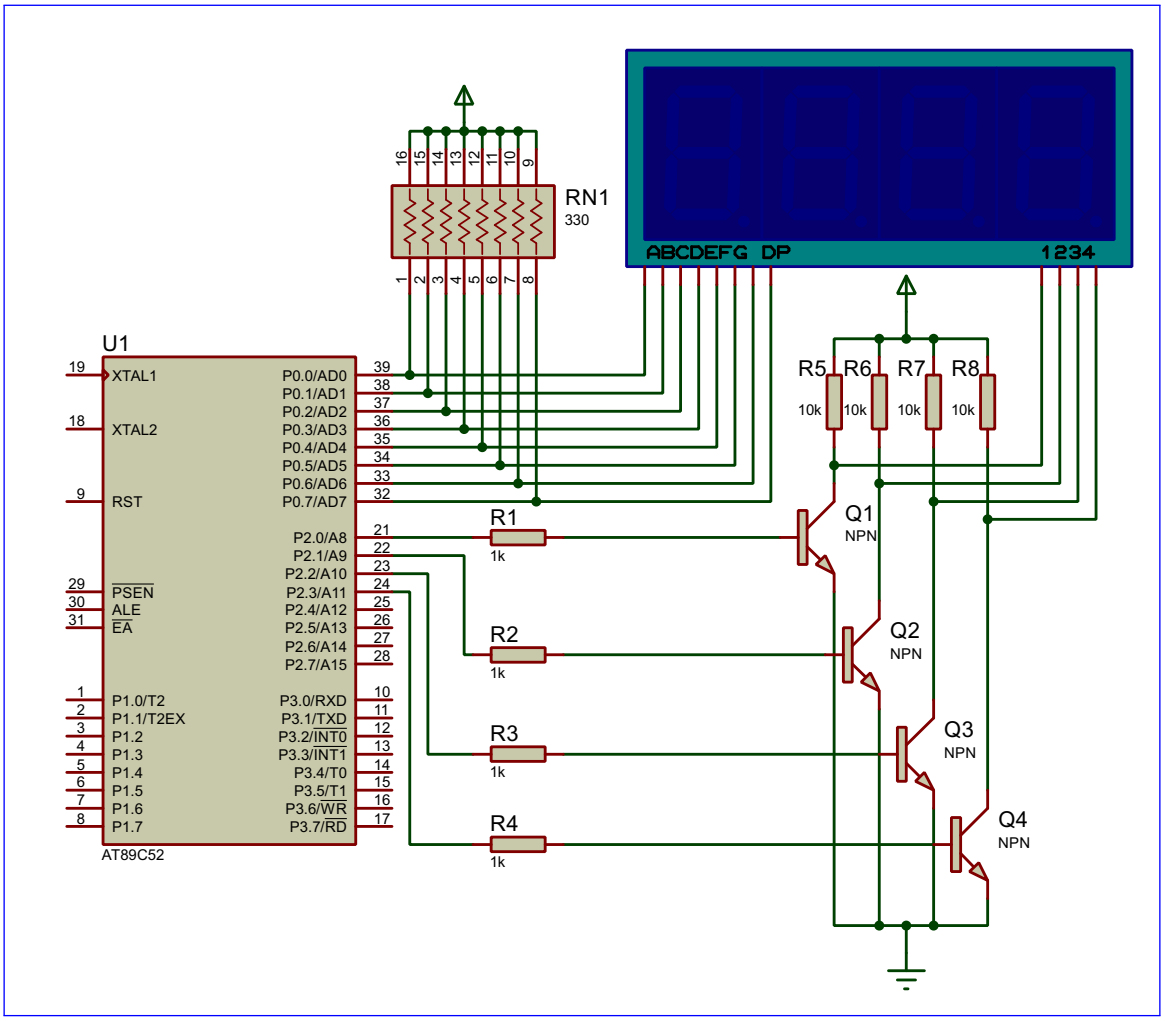

Figure 13: Proteus Schematic Problem no.3

Assembly

```
ORG 00H
MOV TMOD,#10H
MOV P2,#00H
MOV DPTR,#LABEL1
START: MOV R0,#00
MOV R1,#00
LOOP1: MOV R7,#27H
MAIN: MOV A,R0
ACALL HTOD
MOV B,A
ANL A,#0FH
ACALL DISPLAY
SETB P2.3
MOV P0,A
ACALL DELAY_T
CLR P2.3
MOV A,B
ANL A,#0F0H
SWAP A
ACALL DISPLAY
SETB P2.2
MOV P0,A
ACALL DELAY_T
CLR P2.2
MOV A,R1
ACALL HTOD
MOV B,A
ANL A,#0FH
ACALL DISPLAY
ORL A,#80H
SETB P2.1
MOV P0,A
ACALL DELAY_T
CLR P2.1
MOV A,B
ANL A,#0F0H
```

```
SWAP A
ACALL DISPLAY
SETB P2.0
MOV P0,A
ACALL DELAY_T
CLR P2.0
DJNZ R7,MAIN
CJNE R0,#3BH,LESS
INC R1
MOV R0,#0FFH
LESS: INC R0
CJNE R1,#3CH,LOOP1
AJMP START
HTOD: MOV B,#0AH
DIV AB
SWAP A
ADD A,B
RET
DELAY_T: MOV TL1,#3FH
MOV TH1,#0E8H
SETB TR1
AGAIN: JNB TF1, AGAIN
CLR TR1
CLR TF1
RET
DISPLAY: MOVC A,@A+DPTR
RET
LABEL1: DB 3FH
DB 06H
DB 5BH
DB 4FH
DB 66H
DB 6DH
DB 7DH
DB 07H
DB 7FH
DB 6FH
END
```

Code 17: Problem no. 3 Assembly

C language

```
#include <reg51.h>
unsigned char led_pattern[10] = {
    0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d,
    0x7d, 0x07, 0x7f, 0x6f};
void delay(void)
{
    TMOD = 0x10;
    TH1 = 0xE9;
    TL1 = 0x3F;
    TR1 = 1;
    while (!TF1)
        ;
    TR1 = 0;
    TF1 = 0;
}
void display(int min, int sec)
{
    int i, r, led[4];
    led[0] = min / 10;
    led[1] = min % 10;
    led[2] = sec / 10;
    led[3] = sec % 10;
    for (r = 0; r < 39; r++)
    {
        P2 = 0x01;
        for (i = 0; i < 4; i++)
        {
            if (i == 1)
                P0 = led_pattern[led[i]] | 0
    x80;
            else
                P0 = led_pattern[led[i]];
            delay();
            P2 <<= 1;
        }
    }
}
void main(void)
{
    int i, j;
    while (1)
        for (i = 0; i < 60; i++)
            for (j = 0; j < 60; j++)
                display(i, j);
}
```

Code 18: Problem no. 3 C language

OUTPUT: Outputs include some snaps taken during that 60 minute period. here delay is used to make clock more accurate. the delay has ti=o be minimum to avoid flickering of LED.programming is done in binary format and we need to convert binary values to decimal equivalent while displaying the digits in LED unit.it also consist of hexadecimal to decimal converter sub-routine, display sub-routine, LED pattern selection sub-routine for efficient programming. ther is decimal point in between minute and second.

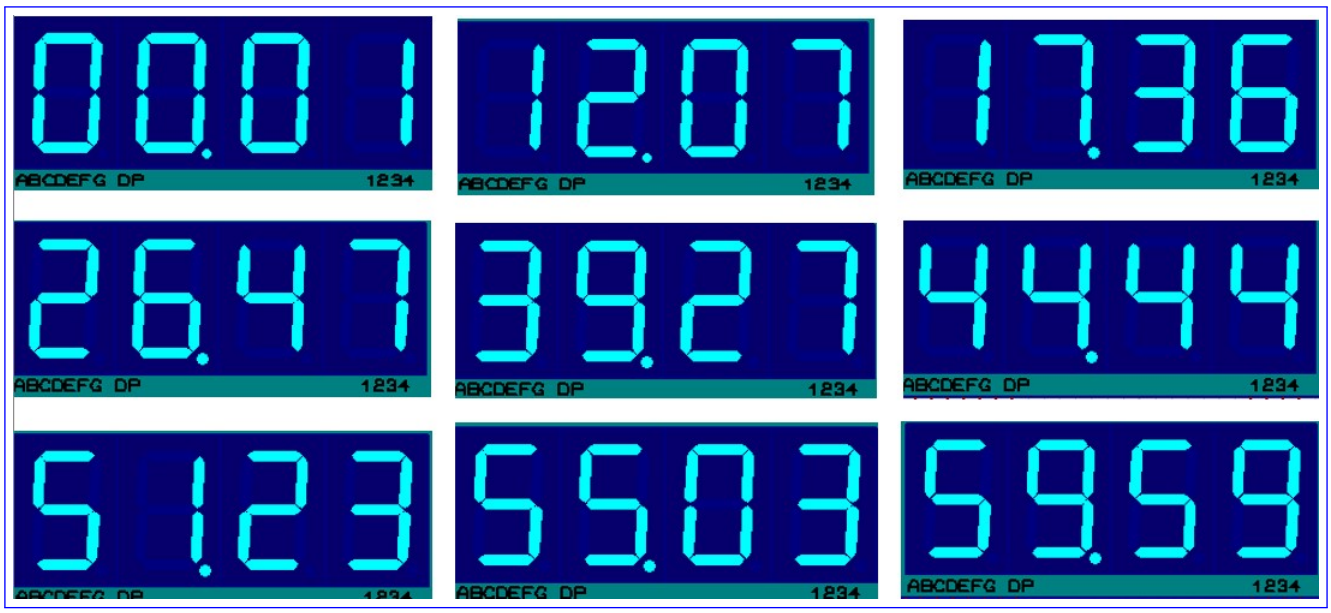

Figure 14: Timer Outputs

5 Conclusion

In this Lab we program timers of 8051/8052 in Microcontroller in C and Assembly Language. we performed various waveform using delay in Timers available in 8051/8052 microcontroller. Kiel IDE is used to generate HEX file from both Assembly and C language and Proteus to simulate the whole circuit.All codes ,schematic and Outputs are included in the report.