Programming Timers of 8051 Microcontroller

Brihat Ratna Bajracharya

Department of Electronics and Computer Engineering, IOE Central Campus, Pulchowk
Lalitpur, Nepal

070bct513@ioe.edu.np

***Abstract*—a computer in a single chip is called microcontroller. All necessary blocks of computer like central processing unit, memory, input and output ports, clock, timers/counters and registers are all embedded into a single chip that is used for various educational and other purposes. Some application using microcontroller need precise and accurate timing delays in their functionality. For this purpose programming 8051 internal timers is necessary. Though the programming is cumbersome and error prone, it is best way to introduce appropriate delays in our system. Internal timers of 8051 has minimum timing delay of 1.085 µs.**

I. INTRODUCTION

The Intel MCS-51 (commonly termed **8051**) is an internally Harvard with CISC (Complex Instruction Set Computing) architecture single chip microcontroller series developed by Intel in 1980 for use in embedded systems. The original MCS-51 family was made using N-type metal-oxide-semiconductor (NMOS) but later versions identified by letter 'C' in their name (e.g. 80C51) used complementary metal-oxide-semiconductor (CMOS) technology.

The 8051 architecture provides many functions (like CPU (Central Processing Unit), RAM (Random Access Memory), ROM (Read Only Memory), I/O (Input/Output), Interrupt logic, Timer, etc.) in a single chip/package.

MCS-51 based microcontrollers typically include one or two UARTs, two or three Timers, 128 or 256 bytes of internal data RAM, 128 bytes of I/O, 512 bytes to 64 kilo-bytes of internal program memory and external data space. The original 8051 runs at 12 MHz clock frequency. Today's 8051 microcontroller has clock frequencies of up to 100 MHz.

A. AT89S52 Micro-controller

The AT89S52 is a low power, high performance CMOS eight bit microcontroller with 8 kilo-bytes of in-system programmable flash memory. The device is manufactured using Atmel's high-density non-volatile memory technology and is compatible with the industry-standard 80C51 instruction set. The Atmel AT89S52 is a powerful microcontroller which provides a highly flexible and cost effective solution to many embedded control applications.

The AT89S52 provides the following standard features:

- 8 kilo-bytes of flash memory,
- 256 bytes of RAM,
- 32 I/O lines,
- Watchdog timer,
- 2 data pointers (DP),
- 3 16-bit timer/counters,
- A six-vector two-level interrupt architecture,
- A full duplex serial port,
- On-chip oscillator, and
- Clock circuitry.

In addition, the AT89S52 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The **idle mode** stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The **power down mode** saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

B. The 8051 Timers

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.

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. They are accessed like any other register like,

```
MOV TL0, #4FH
MOV R5, TH0
```

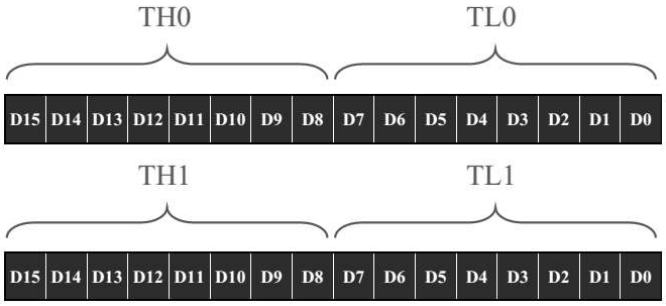


Fig. Timer 0 and Timer 1 showing corresponding high and low byte

1) TMOD (Timer Mode) Register

Both timers 0 and 1 use the same register, called TMOD (timer mode) to set the various timer operation modes. TMOD is a 8-bit register whose lower 4 bits are for Timer 0 and upper 4 bits are for Timer 1. In each case, the lower 2 bits are used to set the timer mode and upper 2 bits to specify the operation.

GATE (Gating Control): Timer/counter is enable when the INTx pin is high and the TRx control pin is set. When cleared, the timer is enabled whenever the TRx control bit is set.

C/T (Counter/Timer): Reset (i.e. 0 or cleared) for timer operation (input is given from internal system clock according to machine cycle of instruction) and set (i.e. 1) for counter operation (input is given from Tx input pin).

Fig. TMOD Register

TABLE – I
TIMER MODES

M1	M0	Mode	Operating Mode
0	0	0	**13-bit timer mode** 8-bit timer/counter 8 bit of THx and least significant 5 bits of TLx are cascaded
0	1	1	**16-bit timer mode** 16-bit timer/counter THx and TLx are cascaded
1	0	2	**8-bit auto reload** 8-bit auto reload timer/counter; THx holds a value which is to be reloaded TLx each time it overflows
1	1	3	**Split timer mode** Only Timer 0 is used in this mode

2) TCON Register

The 8051 microcontroller has one 8-bit register that holds the timer flags, interrupt flags and timer run control bus

Fig. TCON Register

TCON Register is used by both timers as well as interrupts. Upper 4 bits (TCON.7 to TCON.4) is used by timer and lower 4 bits (TCON.3 to TCON.0) is used by interrupts. TCON bits are:

```
TF1 = Timer 1 Overflow Flag
TR1 = Timer 1 Run
TF0 = Timer 0 Overflow Flag
TR0 = Timer 0 Run
IE1 = Interrupt 1 Edge Detect Flag
IT1 = Interrupt 1 Type Control
IE0 = Interrupt 0 Edge Detect Flag
IT0 = Interrupt 0 Type Control
```

Overflow flag is set when timer overflows and edge detect flag is set when interrupt edge is detected. Timer run is set to start the timer and reset to stop the timer. If type control bit is set, it detects falling edge interrupt while if type control bit is reset, it detect zero level interrupt.

3) Clock Source for Timers

Timer operation is selected by making C/T bit of TMOD register zero. The timer is clocked from oscillator. Frequency for timer is $\frac{1}{12}^{th}$ the frequency of the crystal attached to 8051 microcontroller.

Frequency of oscillator = 11.0592 MHz

Machine cycle frequency = $\frac{11.0592\ MHz}{12} = 921.6\ KHz$

Machine cycle period = $\frac{1}{921.6\ KHz} = 1.085\ \mu s$

4) Timer Modes

a. Timer Mode 0 (13 bit timer mode)

This mode is selected by choosing M1 = 0 and M0 = 0 in the TMOD register. TH (8-bit) and TL (lower 5 bits) of timer are cascaded to form a 13-bit timer. Most significant bit of timer is TH7 and least significant bit is TL0. Values from 0000 H to 1FFF H can be loaded into timer's register.

After loading required values to TH and TL register, Timer 0 is started using command SETB TR0 and Timer 1 is started by using the command SETB TR1. Timer counts from loaded values up to 1FFF H and rolls over to 0000 H setting TF0 flag for Timer 0 and TF1 for Timer 1. The process can be repeated by reloading timer registers and resetting timer flag to zero.

b. Timer Mode 1 (16-bit timer mode)

This mode is selected by choosing M1 = 0 and M0 = 1 in the TMOD register. TH (8-bit) and TL (8 bits) of timer are cascaded to form a 13-bit timer. Values from 0000 H to FFFF H can be loaded into timer's register.

After loading required values to TH and TL register, Timer 0 is started using command SETB TR0 and Timer 1 is started by using the command SETB TR1. Timer counts from loaded values up to FFFF H and rolls over to 0000 H setting TF0 flag for Timer 0 and TF1 for Timer 1. The process can be repeated by reloading timer registers and resetting timer flag to zero.

c. Timer Mode 2 (8-bit auto reload)

This mode is selected by choosing M1 = 1 and M0 = 0 in the TMOD register. Only TL (8 bits) register of timer is used. Values from 00 H to FF H can be loaded into timer's register. TH register automatically loads a copy of the initial value into TL register.

After TL receives a copy of the 8-bit initial value, TL0 of Timer 0 is started using command SETB TR0 and TL1 of Timer 1 is started by using the command SETB TR1. Timer counts from loaded values up to FF H and rolls over to 00 H setting TF0 flag for Timer 0 and TF1 for Timer 1. The process can be repeated by resetting timer flag to zero.

d. Timer Mode 3 (Split timer mode)

This mode is selected by choosing M1 = 1 and M0 = 1 in the TMOD register. Timer 0 splits into two 8-bit timers. TL0 and TH0 act as two separate timers. TL0 uses TR0 and TF0 while TH0 uses TR1 and TF1 bit.

After TL0 and/or TH0 are initialized with 8-bit initial value, timers are started. TL0 is started using command SETB TR0 and TH0 is started by using the command SETB TR1. Timer counts from loaded values up to FF H and rolls over to 00 H setting TF0 flag if TL0 overflows and setting TF1 flag if TH0 overflows. The process can be repeated by reloading timer registers and resetting timer flag to zero. Timer 1 cannot be used in mode 3.

C. Seven Segment Display

A seven segment display electronic display device for displaying decimal numbers and a decimal point. The seven segments are arranged as a rectangle of two vertical segments on each side with a horizontal segment on the top, middle, and bottom. The segments of seven segment display are referred to by the letters A to G, where the optional decimal point (an 'eighth' segment, referred by DP) is used for display of non-integer numbers. Construction of seven segment display is done either by connecting all cathodes (negative terminals) or all anodes (positive terminals) of the segment to a common pin and is referred to as a 'common cathode' or common anode device respectively. Seven segment display is used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.

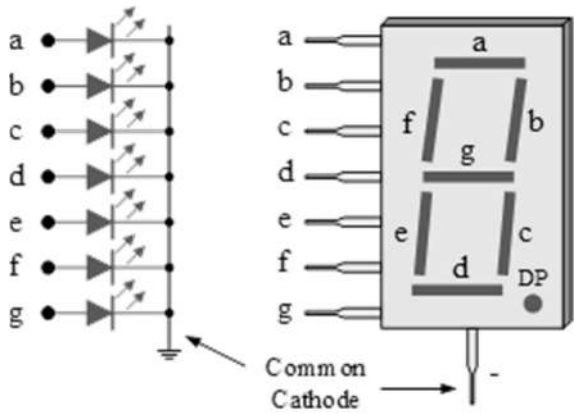


Fig. Common cathode seven segment display

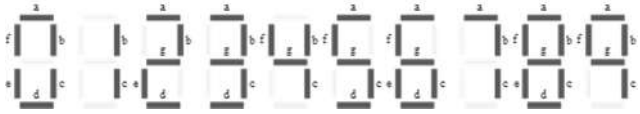
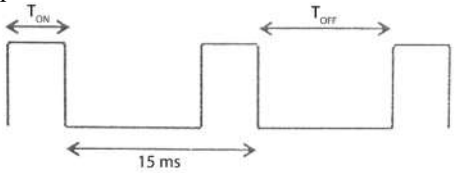


Figure showing display segments for all numbers

II. ACTIVITY I

Generate a periodic square wave having a period of 15 ms and a duty cycle of 20% as shown in figure 10. 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 times.



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

Assembly Code:

```
          ORG 00H

       ;  TIMER 1 IN MODE 0
          MOV TMOD,#00       ;2

HERE:     MOV TL1,#01H       ;2
          MOV TH1,#0AAH      ;2
          SETB P2.0          ;1
          ACALL DELAY        ;2

          MOV R2,#04H        ;2
AGN:      MOV TL1,#01H       ;2
          MOV TH1,#0AAH      ;2
          CLR P2.0           ;1
          ACALL DELAY        ;2
          DJNZ R2,AGN        ;2
          SJMP HERE          ;2

DELAY:    SETB TR1           ;1
AGAIN:    JNB TF1, AGAIN     ;2750
          CLR TR1            ;1
          CLR TF1            ;1
          RET                ;2

          END
```

Discussion:

In this activity, 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 is given by

T_{ON} = duty cycle $\times T_{total} = 20\ \% \times 15\ ms = 3\ ms$ and

$T_{OFF} = T_{total} - T_{ON} = 15\ ms - 3\ ms = 12\ ms$

In above program, 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.

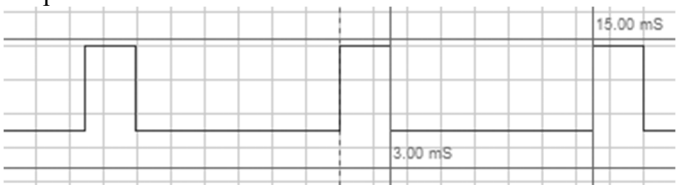
Timer registers are set to AA 01 H (1010 1010 xxx0 0001 B) which is equivalent to 1541 H (0001 0101 0100 0001 B) that gives 1F FF H – 15 41 H = 0A BE H = 2750 machine cycle.

Hence,

Total machine cycle per loop $= 2766 + 4 \times 2764$
$= 13822$ cycles

Time period of square wave $= 13822 \times 1.085\mu s$
$= 14.99$ ms.

Output:



Observation:

From above waveform, it is observed that,

T_{TOTAL} = 15.00 ms

T_{ON} = 3.00 ms and T_{OFF} = 15.00 – 3.00 = 12.00 ms

(b) Using Timer 0 in mode 1 (16-bit timer mode)

Assembly Code:

```
          ORG 00H

        ; TIMER 0 IN MODE 1
          MOV TMOD,#01      ;2

HERE:     MOV TL0,#41H      ;2
          MOV TH0,#0F5H     ;2
          SETB P2.0         ;1
          ACALL DELAY       ;2

          MOV R2,#04H       ;2
AGN:      MOV TL0,#41H      ;2
          MOV TH0,#0F5H     ;2
          CLR P2.0          ;1
          ACALL DELAY       ;2
          DJNZ R2,AGN       ;2
          SJMP HERE         ;2

DELAY:    SETB TR0          ;1
AGAIN:    JNB TF0, AGAIN    ;2750
          CLR TR0           ;1
          CLR TF0           ;1
          RET               ;2

          END
```

Discussion:

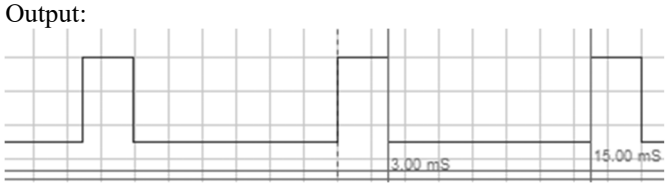
In above program, Timer 1 is used in mode 0. Mode 0 uses 16 bit timer. Hence, timer registers are set to F5 41 H. So machine cycle of delay loop is FF FF H – F5 41 H = 0A BE H or 2750 machine cycle.

Hence,

Total machine cycle per loop $= 2766 + 4 \times 2764$
$= 13822$ cycles

Time period of square wave $= 13822 \times 1.085\mu s$
$= 14.99$ ms.

Output:



Observation:

From above waveform, it is observed that,

T_{TOTAL} = 15.00 ms

T_{ON} = 3.00 ms and T_{OFF} = 15.00 – 3.00 = 12.00 ms

(c) Using Timer 1 in mode 2 (8-bit auto-reload timer mode)

Assembly Code:

```
          ORG 00H

        ; TIMER 1 IN MODE 2
          MOV TMOD,#20H     ;2

HERE:     MOV R2,#10H       ;2
AGN:      MOV TL1,#5EH      ;2
          SETB P2.0         ;1
          ACALL DELAY       ;2
          DJNZ R2,AGN       ;2

          MOV R2,#40H       ;2
AGN1:     MOV TL1,#5EH      ;2
          CLR P2.0          ;1
          ACALL DELAY       ;2
          DJNZ R2,AGN1      ;2
          SJMP HERE         ;2

DELAY:    SETB TR1          ;1
AGAIN:    JNB TF1, AGAIN    ;161
          CLR TR1           ;1
          CLR TF1           ;1
          RET               ;2

          END
```

Discussion:

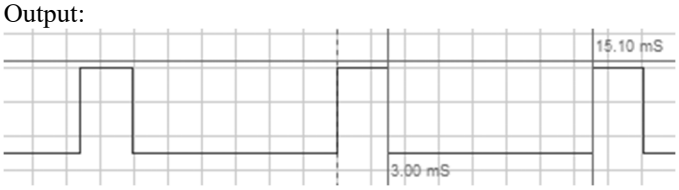
Here, Timer 1 is used in mode 2 for delay purpose. Since mode 2 is 8-bit auto reload timer, we need to loop timer delay to make delay more than 255 machine cycles. In above program timer register is loaded with value 5E H and delay is looped 16 times for on cycle and 64 times for off cycle.

Machine cycle of delay loop is FF H – 5E H = A1 H or 161 machine cycle.

Total machine cycle per loop
$= 2 + 16 \times 171 + 64 \times 173 + 2$
$= 13812$ cycles

Time period of square wave $= 13812 \times 1.085\mu s$
$= 14.98$ ms.

Output:



Observation:

From above waveform, it is observed that,

T_{TOTAL} = 15.10 ms

T_{ON} = 3.00 ms and T_{OFF} = 15.10 – 3.00 = 12.10 ms

(d) Using Timer 0 (TL0) in mode 3 (8-bit split timer mode)

Assembly Code:

```
          ORG 00H

        ; TIMER 0 IN MODE 3
          MOV TMOD,#03H     ;2
```

```
HERE:     MOV R2,#10H       ;2
AGN:      MOV TL0,#5EH      ;2
          SETB P2.0         ;1
          ACALL DELAY       ;2
          DJNZ R2,AGN       ;2

          MOV R2,#40H       ;2
AGN1:     MOV TL0,#5EH      ;2
          CLR P2.0          ;1
          ACALL DELAY       ;2
          DJNZ R2,AGN1      ;2
          SJMP HERE         ;2

DELAY:    SETB TR0          ;1
AGAIN:    JNB TF0, AGAIN    ;161
          CLR TR0           ;1
          CLR TF0           ;1
          RET               ;2

          END
```

Discussion:

Here, Timer 0 is used in mode 3 for delay purpose. Since mode 3 is 8-bit split timer mode very much similar to mode 2, we need repeat the delay loop several times to get required delay. In mode 3, we have used timer register TL0 as our delay timer. In this case, timer is started by setting TR0 bit an timer overflow is indicated by TF0 flag. In above program timer register is loaded with value 5E H and delay is looped 16 times for on cycle and 64 times for off cycle.

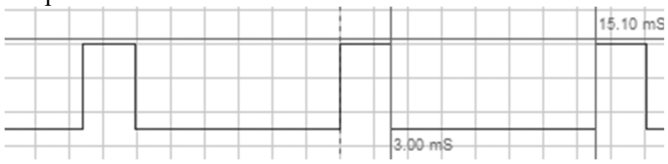
Machine cycle of delay loop is FF H – 5E H = A1 H or 161 machine cycle.

Total machine cycle per loop

$$= 2 + 16 \times 171 + 64 \times 173 + 2$$
$$= 13812 \text{ cycles}$$

Time period of square wave $= 13812 \times 1.085\mu s$
$= 14.98$ ms.

Output:



Observation:

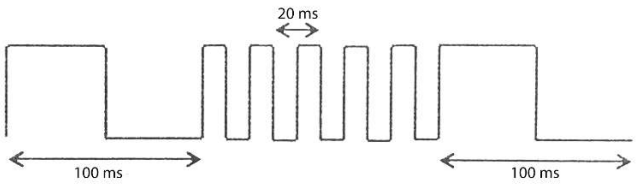
From above waveform, it is observed that,

T_{TOTAL} = 15.10 ms

T_{ON} = 3.00 ms and T_{OFF} = 15.10 – 3.00 = 12.10 ms

III. ACTIVITY II

Generate the periodic waveform as shown in figure. 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.



(a) Using Timer 0 in mode 0 (13-bit timer mode)

Assembly Code:

```
          ORG 00H

        ; TIMER 0 MODE 0
          MOV TMOD,#00      ;2

REPEAT:   MOV R2,#02H       ;2
LOOP1:    MOV R1,#06H       ;2
HERE1:    MOV TL0,#0DH      ;2
          MOV TH0,#10H      ;2
          ACALL DELAY       ;2
          DJNZ R1,HERE1     ;2
          CPL P0.0          ;1
          DJNZ R2,LOOP1     ;2

          MOV R2,#0AH       ;2
LOOP2:    MOV R1,#06H       ;2
HERE2:    MOV TL0,#02H      ;2
          MOV TH0,#0D0H     ;2
          ACALL DELAY       ;2
          DJNZ R1,HERE2     ;2
          CPL P0.0          ;1
          DJNZ R2,LOOP2     ;1
          SJMP REPEAT       ;2

DELAY:    SETB TR0          ;1

; X = 7666 FOR 50 ms AND 1533 FOR 10 ms
AGAIN:    JNB TF0, AGAIN    ;X
          CLR TR0           ;1
          CLR TF0           ;1
          RET               ;2

          END
```

Discussion:

In this activity, there are two waveforms that we have to 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.

Machine cycle required for 50 ms delay $\frac{50\times10^{-3}}{1.085\times10^{-6}} = 46083$ i.e. B4 03 H. And in mode 0, timer register cannot hold more than 1F FF H, adjustment is done by repeating the loop several times. For square wave with time period of 100 ms, 10 0D (0001 0000 0000 1101 B) is loaded in timer register equivalent to 02 0D H which gives 1F FF H – 02 0D H = 1D F2 H or 7666 machine cycle. Similarly for square wave with time period of 20 ms, D0 02 (1101 0000 0000 0010 B) is loaded in timer

register equivalent to 1A 02 H which gives 1F FF H – 1A 02 H = 05 FD H or 1533 machine cycle.

Now,

Total machine cycle in first loop $= 6 \times 7681 = 46086$

Total on time $= 46086 \times 1.085\mu s = 50\ ms$

Total off time $= 50\ ms$ (same as on time)

Similarly,

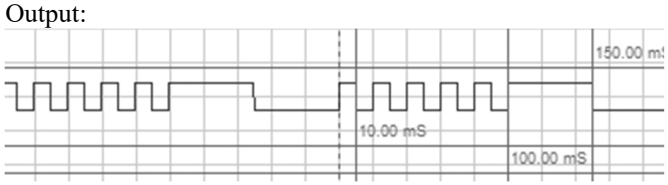
Total machine cycle in second loop $= 6 \times 1546 = 9276$

Total on time $= 9276 \times 1.085\mu s = 10.06\ ms$

Total off time $= 10.06\ ms$ (same as on time)

To make the waveform periodic, second loop is repeated for ten times.

Output:



Observation:

From above waveform, it is observed that,

T_{TOTAL} = 200.00 ms

T_{ON} (1)= 10.00 ms and T_{OFF} (1) = 10.00 ms

T_{ON} (2)= 50.00 ms and T_{OFF} (2) = 50.00 ms

(b) Using Timer 1 in mode 1 (16-bit timer mode)

Assembly Code:

```
            ORG 00H

          ; TIMER 1 MODE 1
            MOV TMOD,#10H      ;2

REPEAT:     MOV R2,#02H        ;2
HERE1:      MOV TL1,#0CH       ;2
            MOV TH1,#4CH       ;2
            ACALL DELAY        ;2
            CPL P0.0           ;1
            DJNZ R2,HERE1      ;2

            MOV R2,#0AH        ;2
HERE2:      MOV TL1,#0FH       ;2
            MOV TH1,#0DCH      ;2
            ACALL DELAY        ;2
            CPL P0.0           ;1
            DJNZ R2,HERE2      ;2
            SJMP REPEAT        ;2

DELAY:      SETB TR1           ;1

; X = 46067 FOR 50 ms AND 9200 FOR 10 ms
AGAIN:      JNB TF1, AGAIN     ;X
            CLR TR1            ;1
            CLR TF1            ;1
            RET                ;2
            END
```

Discussion:

In above program, timer 1 is used in mode 1 which is 16 bit timer mode. Since it can be used for delay upto 65536 machine cycles, required value is loaded in timer mode for required delay directly. Hence for 50 ms delay, value loaded in timer register is 4C 0C H that produce FF FF H – 4C 0C H = B3 F3 H or 46067 machine cycles. Also for 10 ms delay, value loaded in timer register is DC 0F H that produce FF FF – DC 0F H = 23 F0 H or 9200 machine cycles.

Now,

Total machine cycle in first loop $= 46067 + 12 = 46079$

Total on time $= 46079 \times 1.085\mu s = 49.99\ ms$

Total off time $= 49.99\ ms$ (same as on time)

Similarly,

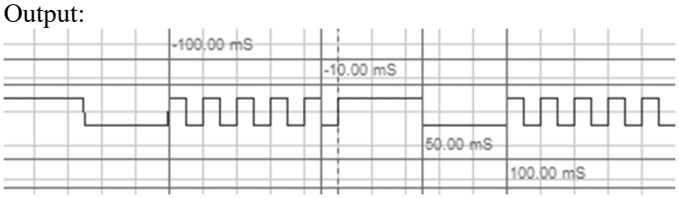
Total machine cycle in second loop $= 9200 + 12 = 9212$

Total on time $= 9212 \times 1.085\mu s = 9.99\ ms$

Total off time $= 9.99\ ms$ (same as on time)

To make the waveform periodic, second loop is repeated for ten times.

Output:



Observation:

From above waveform, it is observed that,

T_{total} = 200.00 ms

T_{ON} (1)= 10.00 ms and T_{OFF} (1) = 10.00 ms

T_{ON} (2)= 50.00 ms and T_{OFF} (2) = 50.00 ms

(c) Using Timer 0 in mode 2 (8-bit auto-reload timer mode)

Assembly Code:

```
            ORG 00H

          ; TIMER 0 MODE 2
            MOV TMOD,#02H      ;2

REPEAT:     MOV R2,#02H        ;2
HERE1:      MOV R1,#0B4H       ;2
HER1:       MOV TL0,#0AH       ;2
            ACALL DELAY        ;2
            DJNZ R1,HER1       ;2
            CPL P0.0           ;1
            DJNZ R2,HERE1      ;2

            MOV R2,#0AH        ;2
HERE2:      MOV R1,#024H       ;2
HER2:       MOV TL0,#0AH       ;2
            ACALL DELAY        ;2
            DJNZ R1,HER2       ;2
            CPL P0.0           ;1
```

```
            DJNZ R2,HERE2      ;2
            SJMP REPEAT        ;2

DELAY:      SETB TR0           ;1
AGAIN:      JNB TF0, AGAIN     ;245
            CLR TR0            ;1
            CLR TF0            ;1
            RET                ;2

            END
```

Discussion:

In above program, timer 0 is used in mode 2 (8-bit auto reload mode). Since it can create delay upto 255 machine cycle only, we split required machine cycle into two part where one part act as value to be loaded to timer register and next part act as multiplier by which delay is to be repeated to get required delay. In above program 0A H is loaded in timer register and it is repeated for B4 H i.e.180 times for 50 ms delay and repeated for 36 times for 10 ms delay. Value loaded in timer register will produce FF H – 0A H = F5 H or 245 machine cycles.

Now,

Total machine cycle in first loop $= 180 \times (245 + 11) + 2$
$= 46082$

Total on time $= 46082 \times 1.085\mu s = 49.99\ ms$

Total off time $= 49.99\ ms$ (same as on time)

Similarly,

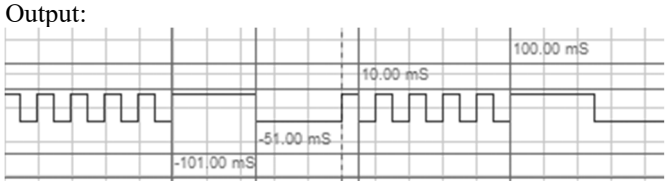
Total machine cycle in second loop $= 36 \times (245 + 11) + 2$
$= 9218$

Total on time $= 9218 \times 1.085\mu s = 10\ ms$

Total off time $= 10\ ms$ (same as on time)

To make the waveform periodic, second loop is repeated for ten times.

Output:



Observation:

From above waveform, it is observed that,

T_{TOTAL} = 200.00 ms

T_{ON} (1)= 10.00 ms and T_{OFF} (1) = 10.00 ms

T_{ON} (2)= 50.00 ms and T_{OFF} (2) = 51.00 ms

(d) Using Timer 0 (TH0) in mode 3 (8-bit split timer mode)

Assembly Code:

```
            ORG 00H

          ; TIMER 0 MODE 3
            MOV TMOD,#03H      ;2

REPEAT:     MOV R2,#02H        ;2
HERE1:      MOV R1,#0B4H       ;2
HER1:       MOV TH0,#09H       ;2
            ACALL DELAY        ;2
            DJNZ R1,HER1       ;2
            CPL P0.0           ;1
            DJNZ R2,HERE1      ;2

            MOV R2,#0AH        ;2
HERE2:      MOV R1,#024H       ;2
HER2:       MOV TH0,#08H       ;2
            ACALL DELAY        ;2
            DJNZ R1,HER2       ;2
            CPL P0.0           ;1
            DJNZ R2,HERE2      ;2
            SJMP REPEAT        ;2

DELAY:      SETB TR1           ;1
AGAIN:      JNB TF1, AGAIN     ;245
            CLR TR1            ;1
            CLR TF1            ;1
            RET                ;2

            END
```

Discussion:

In above program, timer 0 is used in mode 3 (split timer mode). Timer register TH0 is used in above program which is started by setting TR1 bit and overflow is indicated by TF1 flag. Since it can create delay up to 255 machine cycle only, we split required machine cycle into two part where one part act as value to be loaded to timer register and next part act as multiplier by which delay is to be repeated to get required delay. In above program 09 H is loaded in timer register and it is repeated for B4 H i.e.180 times for 50 ms delay and 08 H is loaded in timer register and repeated for 36 times for 10 ms delay. Value loaded in timer register will produce FF H – 0A H = F5 H or 245 machine cycles.

Now,

Total machine cycle in first loop $= 180 \times (245 + 11) + 2$
$= 46082$

Total on time $= 46082 \times 1.085\mu s = 49.99\ ms$

Total off time $= 49.99\ ms$ (same as on time)

Similarly,

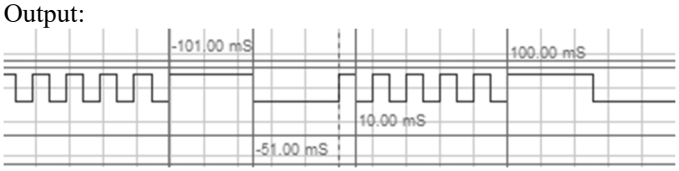
Total machine cycle in second loop $= 36 \times (245 + 11) + 2$
$= 9218$

Total on time $= 9218 \times 1.085\mu s = 10\ ms$

Total off time $= 10\ ms$ (same as on time)

To make the waveform periodic, second loop is repeated for ten times.

Output:



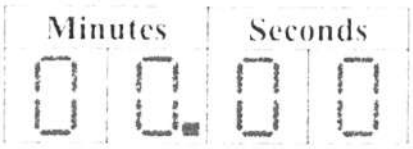
Observation:

From above waveform, it is observed that,

T_{TOTAL} = 200.00 ms
T_{ON} (1)= 10.00 ms and T_{OFF} (1) = 10.00 ms
T_{ON} (2)= 50.00 ms and T_{OFF} (2) = 51.00 ms

IV. Activity III

Design a digital clock that displays **minutes** and **seconds** in double digit format as shown in figure. 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).



Assembly Code:

```
            ORG 00H

            MOV TMOD,#10H           ;2
            MOV P2,#00H             ;2
            MOV DPTR,#LABEL1

START:      MOV R0,#00              ;2
            MOV R1,#00              ;2

; DISPLAY
LOOP1:      MOV R7,#27H             ;2
MAIN:       MOV A,R0                ;1
            ACALL HTOD              ;2
            MOV B,A                 ;1
            ANL A,#0FH              ;2
            ACALL DISPLAY           ;2
            SETB P2.3               ;1
            MOV P0,A                ;1
            ACALL DELAY_T           ;2
            CLR P2.3                ;1

            MOV A,B                 ;1
            ANL A,#0F0H             ;2
            SWAP A                  ;1
            ACALL DISPLAY           ;2
            SETB P2.2               ;1
            MOV P0,A                ;1
            ACALL DELAY_T           ;2
            CLR P2.2                ;1

            MOV A,R1                ;1
            ACALL HTOD              ;2
            MOV B,A                 ;1
            ANL A,#0FH              ;2
            ACALL DISPLAY           ;2
            ORL A,#80H              ;2
            SETB P2.1               ;1
            MOV P0,A                ;1
            ACALL DELAY_T           ;2
            CLR P2.1                ;1

            MOV A,B                 ;1
            ANL A,#0F0H             ;2
            SWAP A                  ;1
            ACALL DISPLAY           ;2
            SETB P2.0               ;1
            MOV P0,A                ;1
            ACALL DELAY_T           ;2
            CLR P2.0                ;1
            DJNZ R7,MAIN            ;2

            CJNE R0,#3BH,LESS       ;3
            INC R1                  ;1
            MOV R0,#0FFH            ;2
LESS:       INC R0                  ;1
            CJNE R1,#3CH,LOOP1      ;3

            AJMP START              ;2

; HEX TO DEC CONVERTER
HTOD:       MOV B,#0AH              ;2
            DIV AB                  ;1
            SWAP A                  ;1
            ADD A,B                 ;1
            RET                     ;2

; DELAY TIMER
DELAY_T:    MOV TL1,#0EH            ;2
            MOV TH1,#0E9H           ;2

            SETB TR1                ;1
AGAIN:      JNB TF1, AGAIN          ; X
            CLR TR1                 ;1
            CLR TF1                 ;1
            RET                     ;2

; CHOOSE REQUIRED PATTERN
DISPLAY:    MOVC A,@A+DPTR          ;1
            RET                     ;2

; LED PATTERNS FOR NUMBERS 0-9
LABEL1:     DB 3FH
            DB 06H
            DB 5BH
            DB 4FH
            DB 66H
            DB 6DH
            DB 7DH
            DB 07H
            DB 7FH
            DB 6FH

            END
```

Discussion:

To make an exact clock, we need to introduce exact delay in our program so that the delay makes the clock run in real time. Also interfacing four seven segment LED units is also very tiresome. We need to create minimal delay between each LED units to avoid flickering display. Also programming is done in binary format and we need to convert binary values to decimal equivalent while displaying the digits in LED unit.

In above program, data (LED pattern) is given via port 0 of 8051 microcontroller and four pins of port 2 are used to activate/deactivate required LED units.

Above program consists of separate hexadecimal to decimal converter sub-routine, display sub-routine, LED pattern selection sub-routine for efficient programming. The flow of above program is as follows:

Timer mode 1 is used for delay purpose being most flexible and easy to program. Two registers R0 and R1 are used to store second and minute value of clock respectively. In each loop, values in registers R0 and R1 are converted to decimal equivalent using `HTOD` sub-routine and displayed on LED unit using `DISLAY` sub-routine. The `DISPLAY` sub-routine internally select required pattern for given number to send to LED unit for display. While displaying the values, masking technique is used to get single digit to be displayed in LED after hexadecimal to decimal conversion.

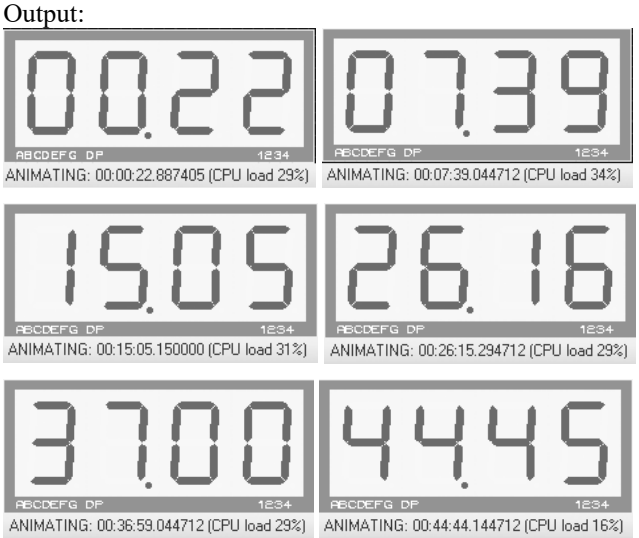
At the end of loop, values in R0 and R1 is compared with 3B H or 59 D as maximum value for second and minute of clock is 59. Once both values reach 59 i.e. time is 59:59, it displays the value and then jumps to the start of program for fresh loop. The clock start from 00:00 from this point.

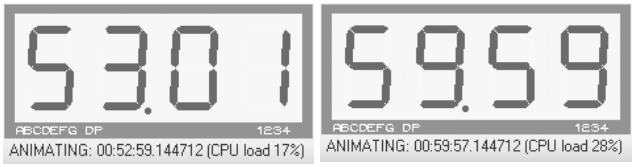
In above program, timer delays are used in between displaying digits in LED units just to avoid flickering display and required delay of the clock is made by repeating display of digits.

Finally, as per the instruction given, minutes and seconds of clock is separated by using a decimal point in between them.

We assumed animating time of Proteus simulation to be real time and coding is done to make clock timing equal to animating time of simulation.

Output:





Observation:

From the simulation done in Proteus, approximately two second delay was observed during time of one hour. This delay can be viewed as the cumulative value of the decimal values we ignored while setting timer registers. This delay can be removed by putting certain delay (equal to delay difference) at the end of loop (after one hour) in above program.

CONCLUSION

Various activities concerned programming internal timers of 8051 microcontroller were done in this lab. Two activities of producing given waveform were done using all four modes of timer in assembly as well as in C programming Language. Keil IDE and Proteus Simulation Software were used to verify the result. Schematic diagrams for each activity made in Proteus are included in Appendix section. Codes to all activities in assembly language are included in this report. In addition, all activities are also done in C programming language and their source code is given in Appendix section.

APPENDIX

Appendix A

Proteus Schematic Capture
For Activity I

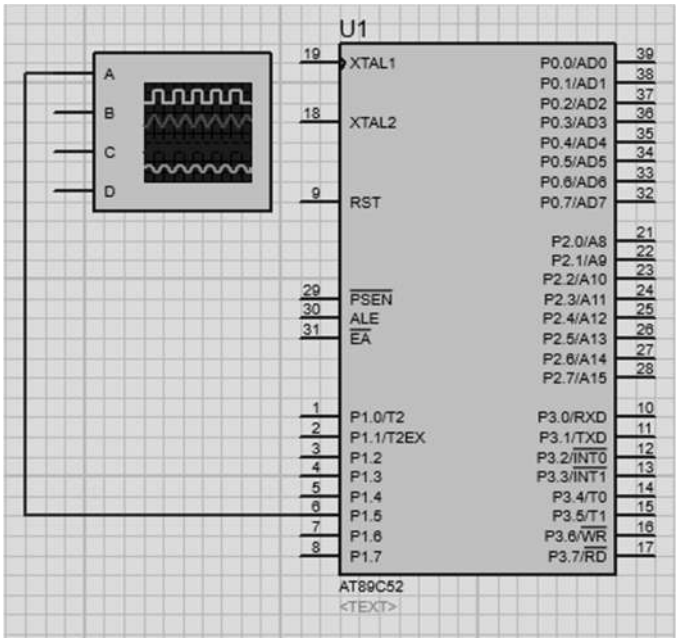


Figure showing circuit diagram in which an oscilloscope is connected to pin P1.5 of 8051 microcontroller to view waveforms of activity I

Appendix B

Proteus Schematic Capture
For Activity II

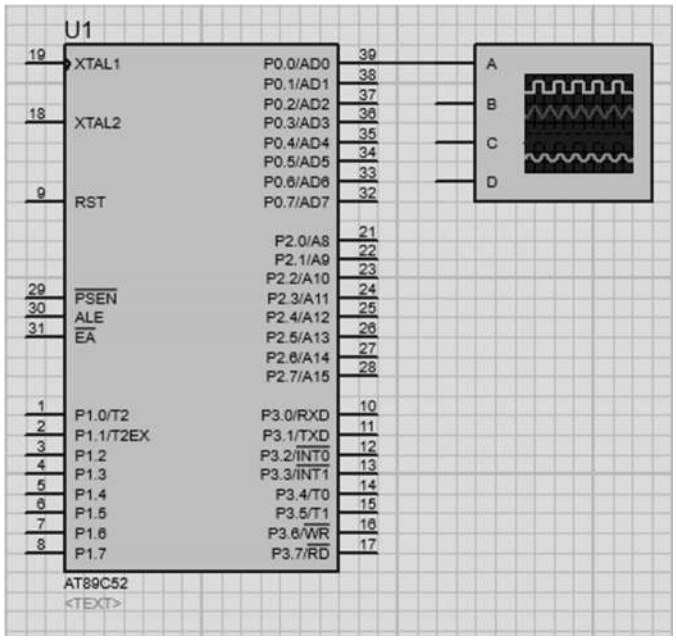


Figure showing circuit diagram in which an oscilloscope is connected to pin P0.0 of 8051 microcontroller to view waveforms of activity II

Appendix C

Proteus Schematic Capture
For Activity III

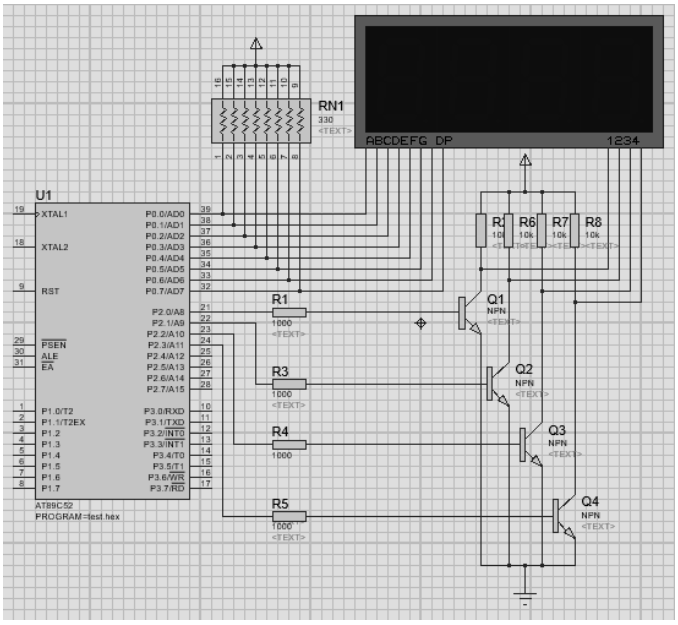


Figure showing circuit diagram for interfacing seven segment display with 8051 microcontroller to display minute and second value of the clock

Appendix D

TABLE – II
SEVEN SEGMENT DISPLAY PATTERN FOR SYMBOLS 0 – 9

Symbol	DP	G	F	E	D	C	B	A	Value
0	0	0	1	1	1	1	1	1	**3F H**
1	0	0	0	0	0	1	1	0	**06 H**
2	0	1	0	1	1	0	1	1	**5B H**
3	0	1	0	0	1	1	1	1	**4F H**
4	0	1	1	0	0	1	1	0	**66 H**
5	0	1	1	0	1	1	0	1	**6D H**
6	0	1	1	1	1	1	0	1	**7D H**
7	0	0	0	0	0	1	1	1	**07 H**
8	0	1	1	1	1	1	1	1	**7F H**
9	0	1	1	0	1	1	1	1	**6F H**

Note: Symbols with decimal point can be obtained by ORing corresponding pattern value with 80 H.

Appendix E

Programs in C programming language

1. C code for Activity I (a)

```
#include <reg51.h>
sbit mybit=P2^0;

void delay(void)
{
  TMOD=0x01;
  TL0=0x32;
  TH0=0xF5;
  TR0=1;
  while (!TF0);
  TR0=0;
  TF0=0;
}

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

2. C code for Activity I (b)

```
#include <reg51.h>
sbit mybit=P2^0;
```

```
void delay(void)
{
  TMOD=0x00;
  TL1=0x12;
  TH1=0xA9;
  TR1=1;
  while (!TF1);
  TR1=0;
  TF1=0;
}

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

3. C code for Activity I (c)

```
#include <reg51.h>
sbit mybit=P2^0;

void delay(void)
{
  int i;
  TMOD=0x20;
  for(i = 0; i < 11; i++)
  {
        TL1=0x04;
        TR1=1;
        while (!TF1);
        TR1=0;
        TF1=0;
  }
}

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

4. C code for Activity I (d)

```
#include <reg51.h>
sbit mybit=P2^0;

void delay(void)
{
  int i;
  TMOD=0x03;
  for(i = 0; i < 11; i++)
  {
        TL0=0x04;
        TR0=1;
        while (!TF0);
        TR0=0;
        TF0=0;
  }
}

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

5. C code for Activity II (a)

```
#include <reg51.h>
sbit mybit=P0^0;

void delay(int factor)
{
  int i;
  TMOD=0x00;

  for(i = 0; i < factor; i++)
  {
        TL0=0x1F;
        TH0=0x6F;
        TR0=1;
        while (!TF0);
        TR0=0;
        TF0=0;
  }
}

void main(void)
{
  char i;
  while (1)
  {
        mybit = 1;
```

```
        delay(10);
        mybit = 0;
        delay(10);
        for(i = 0; i < 5; i++)
        {
              mybit = 1;
              delay(2);
              mybit = 0;
              delay(2);
        }
  };
}
```

6. C code for Activity II (b)

```
#include <reg51.h>
sbit mybit=P0^0;

void delay(int factor)
{
  int i;
  TMOD=0x10;

  for(i = 0; i < factor; i++)
  {
        TL1=0xFF;
        TH1=0xED;
        TR1=1;
        while (!TF1);
        TR1=0;
        TF1=0;
  }
}

void main(void)
{
  char i;
  while (1)
  {
        mybit = 1;
        delay(10);
        mybit = 0;
        delay(10);
        for(i = 0; i < 5; i++)
        {
              mybit = 1;
              delay(2);
              mybit = 0;
              delay(2);
        }
  };
}
```

7. C code for Activity II (c)

```
#include <reg51.h>
sbit mybit=P0^0;

void delay(int factor)
{
  int i;
  for(i = 0; i < 18 * factor; i++)
  {
        TMOD=0x02;
        TL0=0x3F;
        TR0=1;
        while (!TF0);
        TR0=0;
        TF0=0;
  }
}

void main(void)
{
  char i;
  while (1)
  {
        mybit = 1;
        delay(10);
        mybit = 0;
        delay(10);
        for(i = 0; i < 5; i++)
        {
              mybit = 1;
              delay(2);
              mybit = 0;
              delay(2);
        }
  };
}
```

8. C code for Activity II (d)

```
#include <reg51.h>
sbit mybit=P0^0;

void delay(int factor)
{
  int i;
  for(i = 0; i < 18 * factor; i++)
  {
        TMOD=0x03;
        TH0=0x3F;
        TR1=1;
        while (!TF1);
        TR1=0;
        TF1=0;
  }
}

void main(void)
{
  char i;
  while (1)
  {
        mybit = 1;
        delay(10);
        mybit = 0;
        delay(10);
        for(i = 0; i < 5; i++)
        {
```

```
            mybit = 1;
            delay(2);
            mybit = 0;
            delay(2);
        }
  };
}
```

9. C code for Activity III

```
#include <reg51.h>
unsigned char led_pattern[10] = {
0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d,
0x7d, 0x07, 0x7f, 0x6f};

void delay(void)
{
  TMOD=0x01;
  TH0=0xE9;
  TL0=0x0E;
  TR0=1;
  while (!TF0);
  TR0=0;
  TF0=0;
}

void display(int min, int sec)
{
  int i, r, led[5];
  led[1] = min / 10;
  led[2] = min % 10;
  led[3] = sec / 10;
  led[4] = sec % 10;

  for(r = 0; r < 39; r++)
  {
    P2 = 0x01;
    for(i = 1; i < 5; i++)
    {
      if(i == 2)
        P0 = led_pattern[led[i]]
             | 0x80;
      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);
}
```


ACKNOWLEDGMENT

This lab report is prepared as a document for activities done in lab concerned with programming timers of 8051 microcontroller and introducing precise delays in the program. This report is made accurate and professional as far as possible. I would like to express our deepest gratitude to our teacher, Mr. Dinesh Baniya Kshatri, for guiding us in the practical. I am very grateful to the Department of Electronics and Computer Engineering (DoECE) of IOE Central Campus, Pulchowk for arranging such a schedule on our academic side.

REFERENCES

[1] (2016) The Wikipedia website. [Online]. Available: https://en.m.wikipedia.org/wiki/Seven-segment_display

[2] (2016) The Wikipedia website. [Online]. Available: https://en.m.wikipedia.org/wiki/Intel_MCS-51

[3] ATMEL, AT89S52 [portable document]. Available: Offline

[4] (2016) Keil Embedded C Tutorial. [Online], Available: http://www.8051projects.net/wiki/Keil_Embedded_C_Tutorial

[5] (2016) Edsim51 website. [Online]. Available: http://www.edsim51.com/8051Notes/8051/timers.html/

[6] D. Kshatri, Programming Timers of 8051/8052 Microcontroller [handout]. Available: Image form.

[7] C. P. Young, The 8051 Microcontroller and Embedded Systems Using Assembly and C [presentation]. Available: Offline