Interfacing 7-Segment LED Display with 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. Intel first introduced MCS-51 microcontroller in 1980. Today various other vendors like Atmel, Infineon Technologies, NXP, Silicon Laboratories, Texas Instruments, Dallas Semiconductors, ASIX, etc. are manufacturing microcontroller compatible with Intel's MCS-51 that can be used in various embedded systems.**

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. 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.

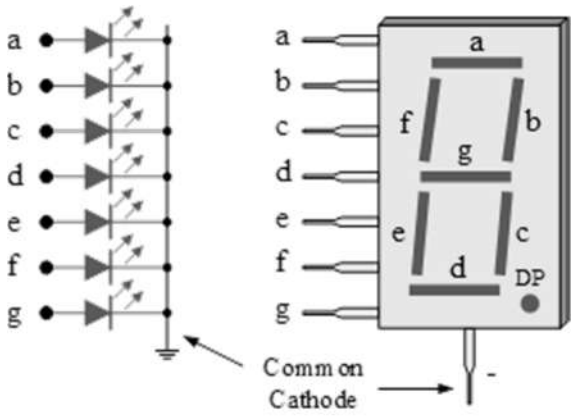


Fig. Common cathode seven segment display

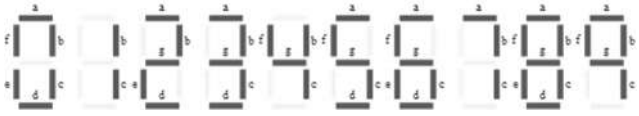


Figure showing display segments for all numbers

II. ACTIVITY I

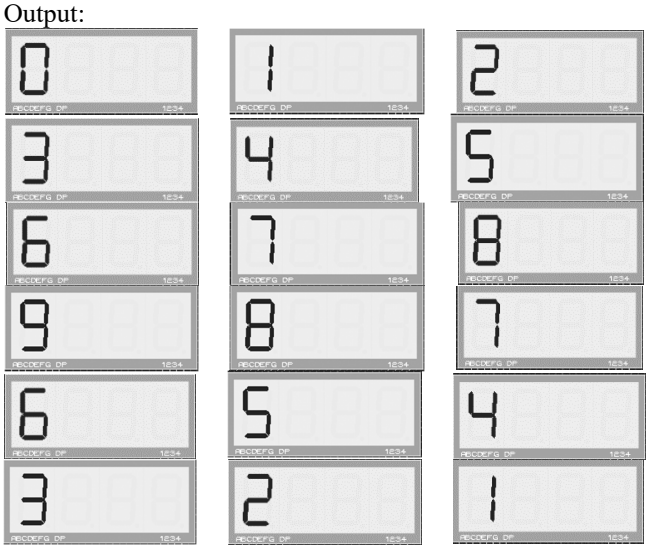
Write code to design a single digit decimal counter. Counter counts up from $(0)_{10}$ to $(9)_{10}$. Once counter reaches the maximum value ($(9)_{10}$) it counts back to $(0)_{10}$. This counting process should repeat indefinitely.

Use a single 7-segment LED unit (non-multiplexed configuration) to display the count value. Use an appropriate timing interval between each count value. Use port zero (P0) of the microcontroller to send the count value to a single 7-segment LED unit. Use pin zero of port two (P2.0) to activate a single 7-segment LED unit.

Assembly Code:

```
            ORG 00H

            MOV 40H,#3FH
            MOV 41H,#06H
            MOV 42H,#5BH
            MOV 43H,#4FH
            MOV 44H,#66H
            MOV 45H,#6DH
            MOV 46H,#7DH
            MOV 47H,#07H
            MOV 48H,#7FH
            MOV 49H,#6FH

            MOV P2,#01H
AGAIN:      MOV R0,#40H
            MOV R2,#0AH
C_INC:      MOV P0,@R0
            INC R0
            ACALL DELAY
            DJNZ R2,C_INC
            DEC R0

            MOV R2,#08H
C_DEC:      DEC R0
            MOV P0,@R0
            ACALL DELAY
            DJNZ R2,C_DEC
            AJMP AGAIN

DELAY:      MOV R3,#5
HERE1:      MOV R4,#255
HERE2:      MOV R5,#255
HERE3:      DJNZ R5,HERE3
            DJNZ R4,HERE2
            DJNZ R3,HERE1
            RET

            END
```

Output:



Discussion:

While interfacing seven segment display for displaying single digit counter, only one port of LED unit is used. So, we can send led pattern for each digit one by one with required delay to show single digit counter in seven segment display. In above program, we have stored display patterns for each digit from memory location 40 H. Also pin zero of port two is made high to activate one LED unit in seven segment display. Then values stored in memory is sent one by one to seven segment display through port zero of 8051 microcontroller with certain delay between two values. Values are sent in such a way that the display shows count from 0 through 9 and then back to 0. This process is repeated in infinite loop for continuous display.

III. ACTIVITY II

Write code to design a double digit decimal counter. Counter counts up from $(00)_{10}$ to $(20)_{10}$. Once counter reaches the maximum value ($(20)_{10}$) it counts back to $(00)_{10}$. This counting process should repeat indefinitely.

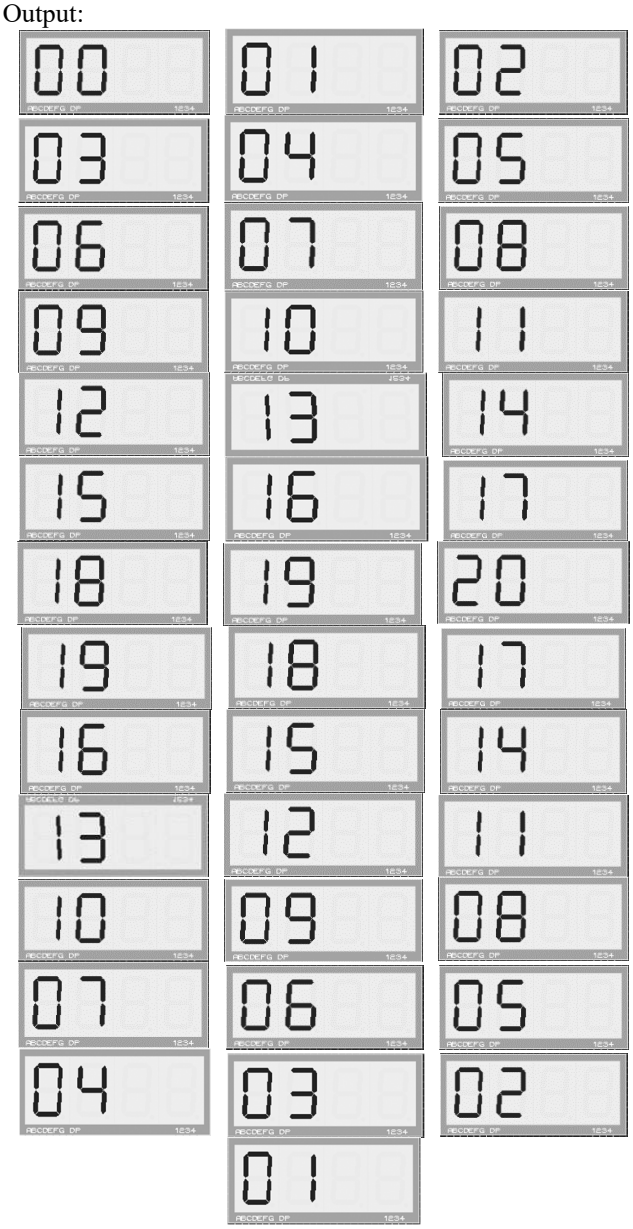
Use two single 7-segment LED units (multiplexed configuration) to display the count value. Use an appropriate timing interval between each count value. Use port zero (P0) of the microcontroller to send the count value to a single 7-segment LED unit. Use pins zero and one of port two (P2.0 and P2.1) to activate two single 7-segment LED units.

Assembly Code:

```
            ORG 00H

            MOV 40H,#3FH
            MOV 41H,#06H
            MOV 42H,#5BH
            MOV 43H,#4FH
            MOV 44H,#66H
            MOV 45H,#6DH
            MOV 46H,#7DH
            MOV 47H,#07H
```

```
            MOV 48H,#7FH
            MOV 49H,#6FH
            MOV 4AH,#3FH

            MOV 50H,40H
            MOV 51H,41H
            MOV 52H,42H

AGAIN:      MOV R1,#50H

            MOV R6,#02H
LOOP2:      MOV R0,#40H
            MOV R5,#0AH
LOOP1:      MOV R7,#255
MAIN:       MOV A,@R1
            MOV P2,#01H
            MOV P0,A
            ACALL DELAY
            MOV A,@R0
            MOV P2,#02H
            MOV P0,A
            ACALL DELAY
            DJNZ R7,MAIN
            INC R0
            DJNZ R5,LOOP1
            INC R1
            DJNZ R6,LOOP2

            MOV R7,#255
LOP:MOV A,@R1
            MOV P2,#01H
            MOV P0,A
            ACALL DELAY
            MOV A,@R0
            MOV P2,#02H
            MOV P0,A
            ACALL DELAY
            DJNZ R7,LOP
            DEC R1

            MOV R6,#02H
LOOP22:     MOV R0,#49H
            MOV R5,#0AH
LOOP11:     MOV R7,#255
MAIN_D:     MOV A,@R1
            MOV P2,#01H
            MOV P0,A
            ACALL DELAY
            MOV A,@R0
            MOV P2,#02H
            MOV P0,A
            ACALL DELAY
            DJNZ R7,MAIN_D
            DEC R0
            DJNZ R5,LOOP11
            DEC R1
            DJNZ R6,LOOP22
            AJMP AGAIN

DELAY:      MOV R3,#02H
DEL1:       MOV R2,#0FAH
DEL2:       DJNZ R2,DEL2
            DJNZ R3,DEL1
            RET

            END
```

Output:



Discussion:

Like previous activity, this activity is also related to decimal counter that runs from 00 through 20. For this, two pins of port two is used to activate two LED units. But due to single data bus (port zero of 8051 microcontroller), we have to interchange value of port two to activate specific LED. In addition, only one seven segment display can be used at a time and we need to display both digits in seven segment display at the same time.

So, to solve this, we need to create an illusion using concept of persistence of vision. Human brain cannot differentiate between two events occurring at a time difference of less than 40 millisecond (ms). Using this concept, we sent two values with delay of less than 40 ms and this alternation is repeated for approximately one second to avoid flickering. Two seven segment display units are turned on and off at appropriate time. Digit patterns are stored in memory location just as in previous activity and count from 00 through 20 and back to 00 is displayed in two seven segment display units. The process is continued infinitely.

IV. Activity III

Write code to display the first (N) numbers of the Fibonacci sequence. The number (N) must be stored in a memory location and can be any integer from $(1)_{10}$ to $(10)_{10}$. Use decimal numbering system to display the sequence. . Use an appropriate timing interval between each sequence value. The sequence should repeat indefinitely.

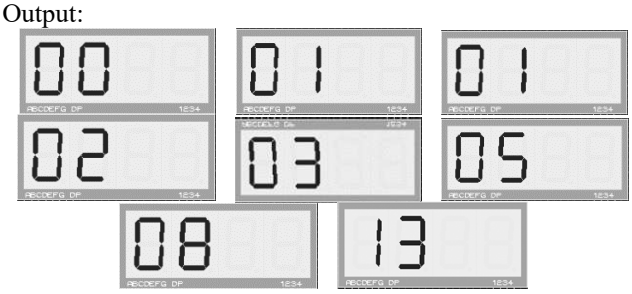
Use port zero (P0) of the microcontroller to send the count value to 7-segment LED units. Use pins of port two to activate required number of 7-segment LED units.

Assembly Code:

```
          ORG 00H

          MOV P2,#00H
          MOV DPTR,#LABEL1
          MOV R0,#50H
          MOV R7,#8
          MOV A,R7
          MOV R6,A

; FIRST TWO TERMS OF FIBONACCI SEQUENCE
          MOV R1,#00H
          MOV R2,#01H
          MOV A,R1
          MOV @R0,A
          INC R0
          DEC R6
          MOV A,R2
          MOV @R0,A
          INC R0
          DEC R6

;CALCULATION OF FIBONACCI TERMS
AGAIN:    MOV A,R1
          ADD A,R2
          MOV @R0,A
          INC R0
          MOV B,R2
          MOV R1,B
          MOV R2,A
          DJNZ R6,AGAIN

;HEX TO DEC CONVERTER
          MOV R0,#50H
          MOV A,R7
          MOV R6,A

AGN2:     MOV A,@R0
          MOV R4,#00H
          MOV B,#0AH
          DIV AB
          MOV R2,A
          SUBB A,#0AH
          JC SKIP
          MOV A,R2
          MOV R3,B
          MOV B,#0AH
          DIV AB
          MOV R4,A
          MOV A,B
          MOV B,R3
SKIP:     MOV A,R2
          SWAP A
          ADD A,B
          MOV B,R4

          MOV @R0,A
          INC R0
          DJNZ R6,AGN2

; DISPLAY
REPEAT:   MOV R0,#50H
          MOV A,R7
          MOV R4,A
LOOP1:    MOV R6,#255
MAIN:     MOV A,@R0
          MOV B,A
          ANL A,#0FH
          MOV P2,#02H
          ACALL DISPLAY
          MOV P0,A
          ACALL DELAY

          MOV A,B
          ANL A,#0F0H
          SWAP A
          MOV P2,#01H
          ACALL DISPLAY
          MOV P0,A
          ACALL DELAY

          DJNZ R6,MAIN
          INC R0
          DJNZ R4,LOOP1
          AJMP REPEAT

DELAY:    MOV R3,#02H
DEL1:     MOV R2,#0FAH
DEL2:     DJNZ R2,DEL2
          DJNZ R3,DEL1
          RET
```

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

;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
```

Output:



Discussion:

Fibonacci sequence is most common sequence of mathematics in which n^{th} term of the sequence is obtained by adding $(n-1)^{th}$ and $(n-2)^{th}$ term. First two terms of Fibonacci sequence is 0 and 1. In this activity, Fibonacci sequence is generated up to required number to terms. And since generated numbers are in hexadecimal number system, it is converted to decimal number system. Then each numbers are displayed one after another with certain delay between r=each number. While displaying, numbers are first brought to accumulator and by proper masking, two digits (upper and lower nibble of accumulator) are sent to two seven segment display. Same concept of persistence of vision is used here to avoid flickering. Display pattern for each digit is stored in memory location and retrieved while necessary. The loop of displaying all numbers of Fibonacci sequence is repeated indefinitely.

V. ACTIVITY IV

Write code to generate the multiplication table of a number (N). The number (N) must be stored in a memory location and can be any integer from $(0)_{10}$ to $(10)_{10}$. Table entries are obtained by multiplying the number (N) with the integers from $(1)_{10}$ to $(10)_{10}$. Use decimal numbering system to display the sequence. . Use an appropriate timing interval between each sequence value. The sequence should repeat indefinitely.

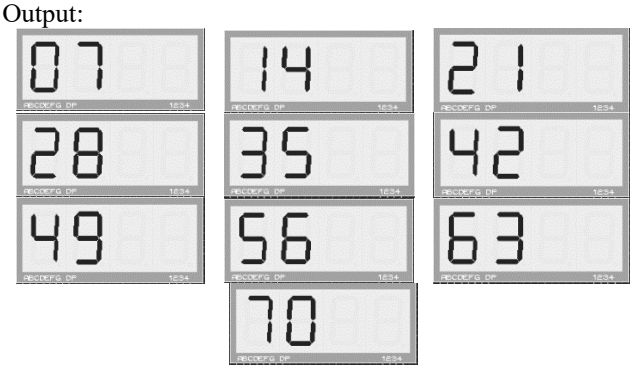
Use port zero (P0) of the microcontroller to send the count value to 7-segment LED units. Use pins of port two to activate required number of 7-segment LED units.

Assembly Code:

```
          ORG 00H

          MOV R7,#7
          MOV P2,#00H
          MOV DPTR,#LABEL1

          MOV B,R7
          MOV R0,#5AH
          MOV R6,#10
AGN:      MOV B,R6
          MOV A,R7
          MUL AB
          MOV @R0,A
          DEC R0
          DJNZ R6,AGN

;HEX TO DEC CONVERTER
          MOV R0,#51H
          MOV R6,#10

AGN2:     MOV A,@R0
          MOV R4,#00H
          MOV B,#0AH
          DIV AB
          MOV R2,A
          SUBB A,#0AH
          JC SKIP
          MOV A,R2
          MOV R3,B
          MOV B,#0AH
          DIV AB
          MOV R4,A
          MOV A,B
          MOV B,R3
          MOV R2,A
SKIP:     MOV A,R2
          SWAP A
          ADD A,B
          MOV B,R4

          MOV @R0,A
          INC R0
          DJNZ R6,AGN2

; DISPLAY
REPEAT:   MOV R0,#51H
          MOV R4,#10
LOOP1:    MOV R7,#255
MAIN:     MOV A,@R0
          MOV B,A
          ANL A,#0FH
          MOV P2,#02H
          ACALL DISPLAY
          MOV P0,A
          ACALL DELAY
          MOV A,B
```

```
            ANL A,#0F0H
            SWAP A
            MOV P2,#01H
            ACALL DISPLAY
            MOV P0,A
            ACALL DELAY
            DJNZ R7,MAIN
            INC R0
            DJNZ R4,LOOP1
            AJMP REPEAT

DELAY:      MOV R3,#02H
DEL1:       MOV R2,#0FAH
DEL2:       DJNZ R2,DEL2
            DJNZ R3,DEL1
            RET

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

;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
```

Output:



Discussion:

This activity displays multiplication table values of a number stored in certain memory location. Like in previous activity, multiplication table values are first calculated and stored in memory. Then it is converted to decimal number system. Then by proper masking of values upper and lower nibble of the memory value is displayed in two seven segment display unit separately. Certain delay is added between two numbers while displaying. Display pattern for each digit is stored in certain location and flickering in seven segment display is addressed in similar way as in previous activity.

VI. Activity V

Write code to display the roll numbers of your lab group members one by one in static format. Each student roll number should be of four characters. Roll numbers begin with a (C) followed by three digits. Display of student roll numbers should repeat indefinitely. Use four 7-segment units (multiplexed configuration) to display a roll number.

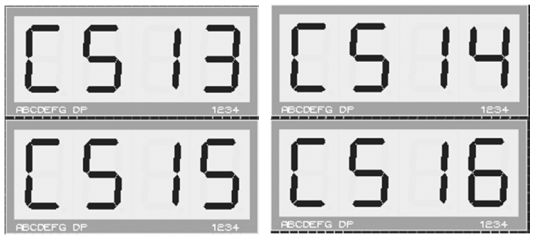
Use port zero (P0) of the microcontroller to send a roll number to the four 7-segment LED units. Use pin zero, one, two, and three of port two (P2.0, P2.1, P2.2, and P2.3) to activate four 7-segment LED units.

Assembly Code:

```
            ORG 00H

;C513
            MOV 40H,#39H
            MOV 41H,#6DH
            MOV 42H,#06H
            MOV 43H,#4FH
;C514
            MOV 44H,#39H
            MOV 45H,#6DH
            MOV 46H,#06H
            MOV 47H,#66H
;C515
            MOV 48H,#39H
            MOV 49H,#6DH
            MOV 4AH,#06H
            MOV 4BH,#6DH
;C516
            MOV 4CH,#39H
            MOV 4DH,#6DH
            MOV 4EH,#06H
            MOV 4FH,#7DH

; DISPLAY
REPEAT:     MOV R0,#40H
            MOV R4,#4
LOOP1:      MOV R7,#255
MAIN:       MOV A,@R0
            SETB P2.0
            MOV P0,A
            ACALL DELAY
            CLR P2.0
            INC R0

            MOV A,@R0
            SETB P2.1
            MOV P0,A
            ACALL DELAY
            CLR P2.1
            INC R0
```

```
            MOV A,@R0
            SETB P2.2
            MOV P0,A
            ACALL DELAY
            CLR P2.2
            INC R0

            MOV A,@R0
            SETB P2.3
            MOV P0,A
            ACALL DELAY
            CLR P2.3

            DEC R0
            DEC R0
            DEC R0

            DJNZ R7,MAIN

            INC R0
            INC R0
            INC R0
            INC R0

            DJNZ R4,LOOP1
            AJMP REPEAT

DELAY:      MOV R3,#02H
DEL1:       MOV R2,#0FAH
DEL2:       DJNZ R2,DEL2
            DJNZ R3,DEL1
            RET

            END
```

Output:



Discussion:

In this activity, we have to display roll numbers (roll number consist of initial of class, C for computer and E for electronics, plus class roll number) of all members of our lab group in static format. This require all four seven segment display units to be used for displaying the roll number. In above assembly program, we have stored display patterns for roll numbers of our lab group member in memory location starting from 40 H. Pins zero, one, two, and three of port two is used to turn on or off four seven segment display units. For each roll number (corresponds to four successive memory values), each memory values are sent via port zero of 8051 microcontroller to display in each of four display segment. Selection of required port of seven segment display is done by providing required value in port two of 8051 microcontroller. In each loop, memory location is increased by four. All four roll numbers are displayed in infinite loop as per question.

VII. Activity VI

Write code to display the roll numbers of your lab group members in scrolling format. Roll numbers should be scrolled towards the left. Roll numbers should be separated using a decimal point. Each student roll number should be of four characters. Roll numbers begin with a (C) followed by three digits. Scrolling process should repeat indefinitely. Use four 7-segment units (multiplexed configuration) to display a roll number. Use an appropriate timing interval while scrolling the digits of each roll number.

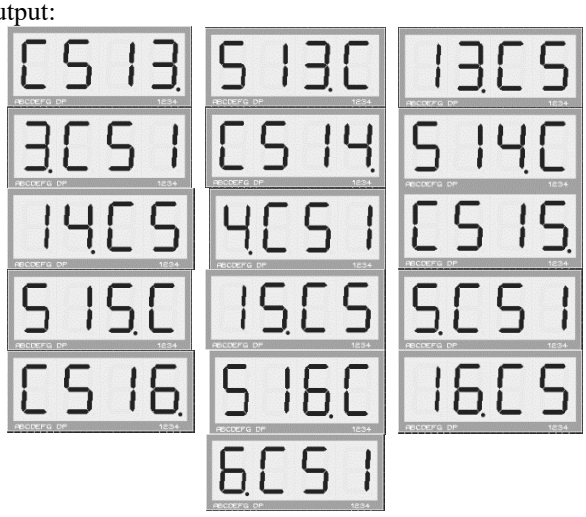
Use port zero (P0) of the microcontroller to send a roll number to the four 7-segment LED units. Use pin zero, one, two, and three of port two (P2.0, P2.1, P2.2, and P2.3) to activate four 7-segment LED units.

Assembly Code:

```
            ORG 00H

;C513
            MOV 40H,#39H
            MOV 41H,#6DH
            MOV 42H,#06H
            MOV 43H,#0CFH
;C514
            MOV 44H,#39H
            MOV 45H,#6DH
            MOV 46H,#06H
            MOV 47H,#0E6H
;C515
            MOV 48H,#39H
            MOV 49H,#6DH
            MOV 4AH,#06H
            MOV 4BH,#0EDH
;C516
            MOV 4CH,#39H
            MOV 4DH,#6DH
            MOV 4EH,#06H
            MOV 4FH,#0FDH
;C51
            MOV 50H,#39H
            MOV 51H,#6DH
            MOV 52H,#06H


; DISPLAY
REPEAT:     MOV R0,#40H
            MOV R4,#10H
LOOP1:      MOV R7,#255
MAIN:       MOV A,@R0
            SETB P2.0
            MOV P0,A
            ACALL DELAY
```

```
            CLR P2.0
            INC R0

            MOV A,@R0
            SETB P2.1
            MOV P0,A
            ACALL DELAY
            CLR P2.1
            INC R0

            MOV A,@R0
            SETB P2.2
            MOV P0,A
            ACALL DELAY
            CLR P2.2
            INC R0

            MOV A,@R0
            SETB P2.3
            MOV P0,A
            ACALL DELAY
            CLR P2.3

            DEC R0
            DEC R0
            DEC R0

            DJNZ R7,MAIN

            INC R0
            DJNZ R4,LOOP1
            AJMP REPEAT

DELAY:      MOV R3,#02H
DEL1:       MOV R2,#0FAH
DEL2:       DJNZ R2,DEL2
            DJNZ R3,DEL1
            RET

            END
```

Output:



Discussion:

This lab is similar to previous activity. Instead of showing roll numbers of lab group members in static format, we have to show it in scrolling format. So first of all, display pattern for roll number is stored in memory location staring from 40 H. as in previous activity, pins zero, one, two, and three of port two is used to turn on or off four seven segment display units. For each roll number (corresponds to four successive memory values), each memory values are sent via port zero of 8051 microcontroller to display in each of four display segment. Selection of required port of seven segment display is done by providing required value in port two of 8051 microcontroller. Unlike previous activity, memory location in this case is increased by one after completion of each loop. All four roll numbers are displayed in infinite loop as per question.

CONCLUSION

Various activities concerned with interfacing seven segment display with 8051 microcontroller were done in this lab. All lab activities were done in assembly as well as in C programming Language. Keil IDE and Proteus Simulation Software were used to verify the result. Schematic diagram made in Proteus is 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

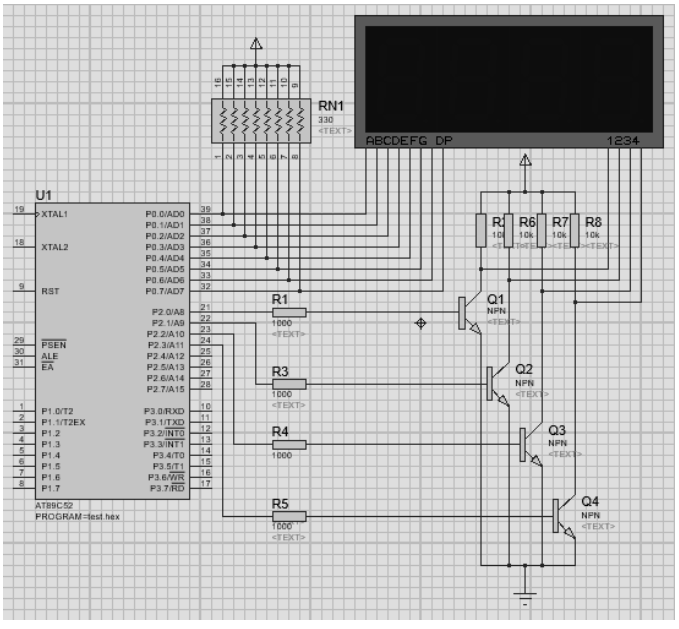


Figure showing circuit diagram for interfacing seven segment display with 8051 microcontroller

Appendix B

TABLE – I
SEVEN SEGMENT DISPLAY PATTERN FOR SYMBOLS 0 – 9, C

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**
C	0	0	1	1	1	0	0	1	**39 H**

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

Appendix C

Programs in C programming language

1. C code for Activity I

```
#include <reg51.h>

unsigned char led_pattern[10] = {
0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d,
0x7d, 0x07, 0x7f, 0x6f};

void delay(int time)
{
  unsigned int i,j;
  for (i=0;i<time;i++)
        for (j=0;j<125;j++);
}

void display(int i)
{
  P0 = led_pattern[i];
  delay(1000);
}

void main(void)
{
  unsigned int i;
  P2 = 0x01;

  while(1)
  {
        for(i=0; i<10; i++)
              display(i);
        for(i=8; i>0; i--)
              display(i);
  }
}
```

2. C code for Activity II

```
#include <reg51.h>

unsigned char led_pattern[10] = {
0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d,
0x7d, 0x07, 0x7f, 0x6f};

void delay(int time)
{
  unsigned int i,j;
  for (i=0;i<time;i++)
        for (j=0;j<125;j++);
}

void display(unsigned int i)
{
  unsigned int j, led1, led2;
  led1 = i / 10;
  led2 = i % 10;
  for(j=0; j<10; j++)
  {
        P2 = 0x1;
        P0 = led_pattern[led1];
        delay(40);

        P2 = 0x2;
        P0 = led_pattern[led2];
        delay(40);
  }
}

void main(void)
{
  unsigned int i;
  while(1)
  {
        for(i=0; i<20; i++)
              display(i);
        for(i=20; i>0; i--)
              display(i);
  }
}
```

3. C code for Activity III

```
#include <reg51.h>
#define N     10

unsigned char led_pattern[10] = {
0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d,
0x7d, 0x07, 0x7f, 0x6f};

void delay(int time)
{
  unsigned int i,j;
  for (i=0;i<time;i++)
        for (j=0;j<125;j++);
```

```
}

void display(unsigned int i)
{
  unsigned int j, led1, led2;
  led1 = i / 10;
  led2 = i % 10;
  for(j=0; j<10; j++)
  {
        P2 = 0x1;
        P0 = led_pattern[led1];
        delay(40);

        P2 = 0x2;
        P0 = led_pattern[led2];
        delay(40);
  }
}

void main(void)
{
  unsigned int i, fibo_seq[N]={0, 1};
  for(i=2; i<N; i++)
        fibo_seq[i] = fibo_seq[i-1] +
        fibo_seq[i-2];
  while(1)
        for(i=0; i<N; i++)
              display(fibo_seq[i]);
}
```

4. C code for Activity IV

```
#include <reg51.h>
#define N     7

unsigned char led_pattern[10] = {
0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d,
0x7d, 0x07, 0x7f, 0x6f};

void delay(int time)
{
  unsigned int i,j;
  for (i=0;i<time;i++)
        for (j=0;j<125;j++);
}

void display(unsigned int i)
{
  unsigned int j;
  for(j=0; j<15; j++)
  {
        P2 = 0x1;
        P0 = led_pattern[i / 10];
        delay(40);

        P2 = 0x2;
        P0 = led_pattern[i % 10];
        delay(40);
  }
}

void main(void)
{
  unsigned int i;
  while(1)
        for(i=1; i<=10; i++)
              display(N*i);
}
```

5. C code for Activity V

```
#include <reg51.h>

unsigned char led_pattern[10] = {
0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d,
0x7d, 0x07, 0x7f, 0x6f};
unsigned char dept_init = 0x39;

void delay(int time)
{
  unsigned int i,j;
  for (i=0; i<time; i++)
        for (j=0; j<125; j++);
}

void display(unsigned int i)
{
  unsigned int j, led2, led3, led4;

  led2 = i / 100;
  led3 = (i - led2 * 100) / 10;
  led4 = i - led2 * 100 - led3 * 10;

  for(j=0; j<20; j++)
  {
        P2 = 0x1;
        P0 = dept_init;
        delay(10);

        P2 = 0x2;
        P0 = led_pattern[led2];
        delay(10);

        P2 = 0x4;
        P0 = led_pattern[led3];
        delay(10);

        P2 = 0x8;
        P0 = led_pattern[led4];
        delay(10);
        }
}

void main(void)
{
  unsigned int i;
  unsigned int roll_no[4] = {513,
514, 515, 516};
```

```
  while(1)
        for(i=0; i<4; i++)
              display(roll_no[i]);
}
```

6. C code for Activity VI

```
#include <reg51.h>

unsigned char scroll_pattern[] = {
0x39, 0x6d, 0x06, 0xcf, 0x39, 0x6d,
0x06, 0xe6, 0x39, 0x6d, 0x06, 0xed,
0x39, 0x6d, 0x06, 0xfd, 0x39, 0x6d,
0x06};

void delay(int time)
{
  unsigned int i,j;
  for (i=0; i<time; i++)
        for (j=0; j<125; j++);
}

void display(unsigned int i)
{
  unsigned int j;
  for(j=0; j<20; j++)
  {
        P2 = 0x1;
        P0 = scroll_pattern[i-4];
        delay(10);

        P2 = 0x2;
        P0 = scroll_pattern[i-3];
        delay(10);

        P2 = 0x4;
        P0 = scroll_pattern[i-2];
        delay(10);

        P2 = 0x8;
        P0 = scroll_pattern[i-1];
        delay(10);
  }
}

void main(void)
{
  unsigned int i;
  while(1)
        for(i=4; i<20; i++)
              display(i);
}
```


ACKNOWLEDGMENT

This lab report is prepared as a document for activities done in lab concerned with interfacing seven segment display with 8051 microcontroller. 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) Electronics Tutorial website. [Online]. Available: http//www.electronics-tutorial.ws/blog/7-segment-display-tutorial.html

[6] D. Kshatri, Interfacing 7-Segment LED Display with 8051/8052 Microcontroller [handout]. Available: Printed form.