Interfacing 7-Segment LED Display with 8051/8052 Micro-controller

Lab Exercises on October 9, 2020

Department of Electronics and Computer Engineering
Pulchowk Campus, Lalitpur

Ashlesh Pandey

PUL074BEX007

Contents

List of Figures

List of Tables

Source Codes

1 Introduction

1.1 MCS-51 Family Microcontroller Chips

Based on the Harvard architecture of designing ICs, the MCS-51 microcontroller chips were originally developed by Intel to be used in small embedded systems. The MCS-51 chips now use Complementary Metal-Oxide Semiconductor (CMOS) instead of the original NMOS, and are thus known as 80C51 chips. Texas Instruments, Atmel, Dallas Semiconductors, Silicon Laboratories, ASTX, and many more distributors manufacture and sell the MCS-51 family microcontroller chips.
The different features of the 8051 microcontroller are:

- 8-bit ALU, Accumulator and Registers, making it an 8-bit microcontroller.
- 4KB of ROM for the programs, also called program memory.
- 8-bit data bus meaning that it can access 8 bits of data in one operation.
- 128 Bytes of RAM for the variables, also called data memory.
- 32 I/O lines, i.e. 4 ports with 8 lines each.
- 16-bit address bus meaning that it can access 65536 locations of RAM and ROM.
- 2 16-bit timers/counters.
- 1 full-duplex serial port for serial communication (UART).
- 6 interrupt sources(2 external interrupts, 2 timer interrupts & 2 serial interrupts).

1.2 7-Segment LED Display

7-segment LED display is a basic electronic device that is made up of 7 led segments each arranged in such a way that specific configurations of these LEDs allow certain alpha-numeric characters to be displayed on the device. Most 7-segment LEDs also contain an additional LED to indicate the decimal point when two or more 7-segment LED displays are used in conjuction. Each of the seven LEDs have a positional reference with pin outs generally being named as a, b, c, d, e, f, g and DP. One end of all the LEDs are commoned out whereas other end is provided with a proper biasing voltage to either turn on or off the segment depending on the terminal polarity. Forward biasing of the appropriate LED segments are used to display the desired character or pattern. Widely used in digital clocks, small-scale calculators, electronic meters and digital display units, the 7-segment LED displays serve a handy purpose in digital electronics.
Sice multiplexing 7-segment displays allows for less number of port connections, less power consumptions and more interfaces, multiplexed 7-segment LED displays are widely used. [1] addresses the multiplexed 7-segment display interfacing with a 8051 microcontroller.

1.2.1 Types of 7-Segment LED Display

Common-Cathode

7-segment LED display where the cathode terminal of the LEDs is made common is called a common cathode (CC) 7-segment LED display. The common cathode terminal is connected to a LOW logic where as individual anode terminals are connected to the HIGH logic through a current limiting resistor as necessary. Certain illumination combinations of the LED segments display the desired character.

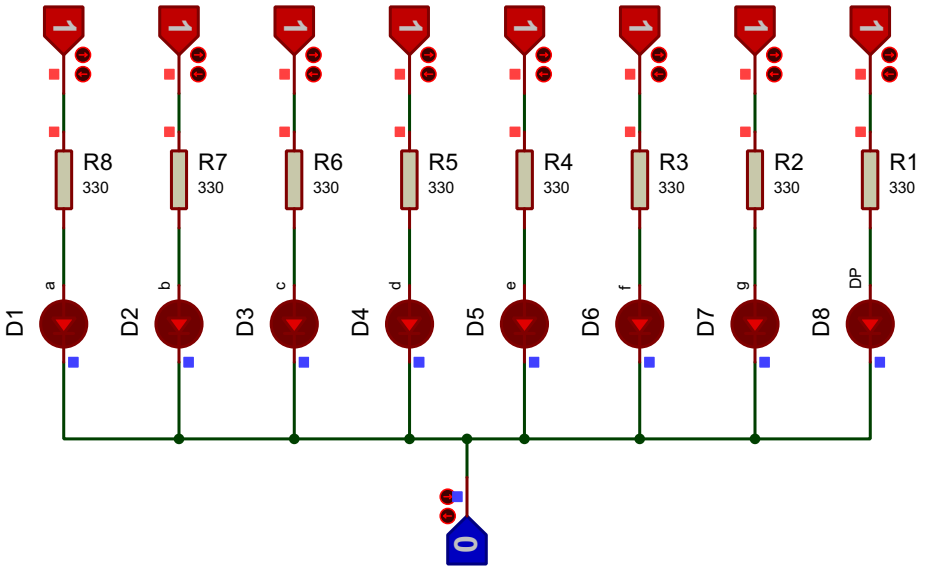


Figure 1: Common-Cathode (CC) configuration for a 7-segment display

Common-Anode

7-segment LED display where the anode terminal of the LEDs is made common is called a common anode (CA) 7-segment LED display. The common anode terminal is connected to a HIGH logic where as individual cathode terminals are connected to the LOW logic through a current limiting resistor as necessary. Certain illumination combinations of the LED segments display the desired character.

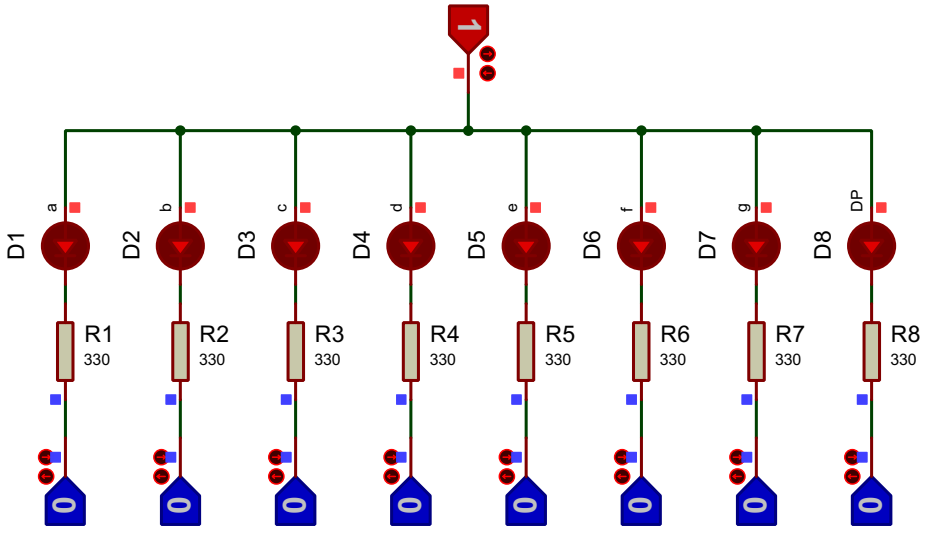


Figure 2: Common-Anode (CA) configuration for a 7-segment display

1.2.2 Digital Drive Pattern

For a common cathode 7-segment LED display, the segments a through g and DP must be provided HIGH logic in order to be turned on. Likewise, for a common anode 7-segment LED display, the segments a through g and DP must be provided LOW logic in order to be turned on. The different combinations that are to

be used to display certain characters are called digital drive pattern. The table that stores these patterns is called a lookup table.

Character	DP	g	f	e	d	c	b	a	HEX 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
E	0	1	1	1	1	0	0	1	79 H

Note: For displaying any character along with a decimal point, the HEX value should be ORed with 80 H.

Table 1: Lookup table for a Common-Cathode 7-segment display

Character	DP	g	f	e	d	c	b	a	HEX value
0	1	1	0	0	0	0	0	0	C0 H
1	1	1	1	1	1	0	0	1	F9 H
2	1	0	1	0	0	1	0	0	A4 H
3	1	0	1	1	0	0	0	0	B0 H
4	1	0	0	1	1	0	0	1	99 H
5	1	0	0	1	0	0	1	0	92 H
6	1	0	0	0	0	0	1	0	82 H
7	1	1	1	1	1	0	0	0	F8 H
8	1	0	0	0	0	0	0	0	80 H
9	1	0	0	1	0	0	0	0	90 H
E	1	0	0	0	0	1	1	0	86 H

Note: For displaying any character along with a decimal point, the HEX value should be ANDed with 7F H.

Table 2: Lookup table for a Common-Anode 7-segment display

2 Objectives

The primary objective of this lab experiment is to understand the various steps involved while interfacing a 7-segment LED display with the 8051/8052 microcontroller. The interfacing knowledge will enable us to write assembly and Embedded C code for the 8051/8052 microcontroller capable of:

- Displaying non-multiplexed and multiplexed output on the 7-segment LED units.
- Displaying static and scrolling outputs on the 7-segment LED units.

3 Lab Experiment Enviornment

The lab experiments will be performed virtually via various simulation softwares. The basic usages of these tools will allow us to visualize and tally the interfacing of a 7-segment LED display with the 8051/8052 microcontroller. Due to a simulated environment, the observations are carefully selected to represent the problems with maximum efficiency.

3.1 Circuit Simulation

A profession PCB layout, circuit design and simulation tool, Proteus Design Suite was used to simulate the interface between a 8052 microcontroller and a common cathode 7-segment LED display. Additional electronic components such as resistors, resistor bus, transistors were also used to attain the circuit shown in Figure 3.

3.2 Code Editor and Compiler

KEIL μVision, which is a product of the ARM Ltd. was used as the code editor for the assembly and embeddded C codes for the different lab problems. KEIL products include C/C++ compilers, debuggers, integrated development and simulation environments, RTOS and middleware libraries, and evaluation boards for ARM, Cortex-M, Cortex-R4, 8051, C166, and 251 processor families. The compiler built-in with KEIL converts the codes into respective HEX codes that are understandable by the microcontroller.

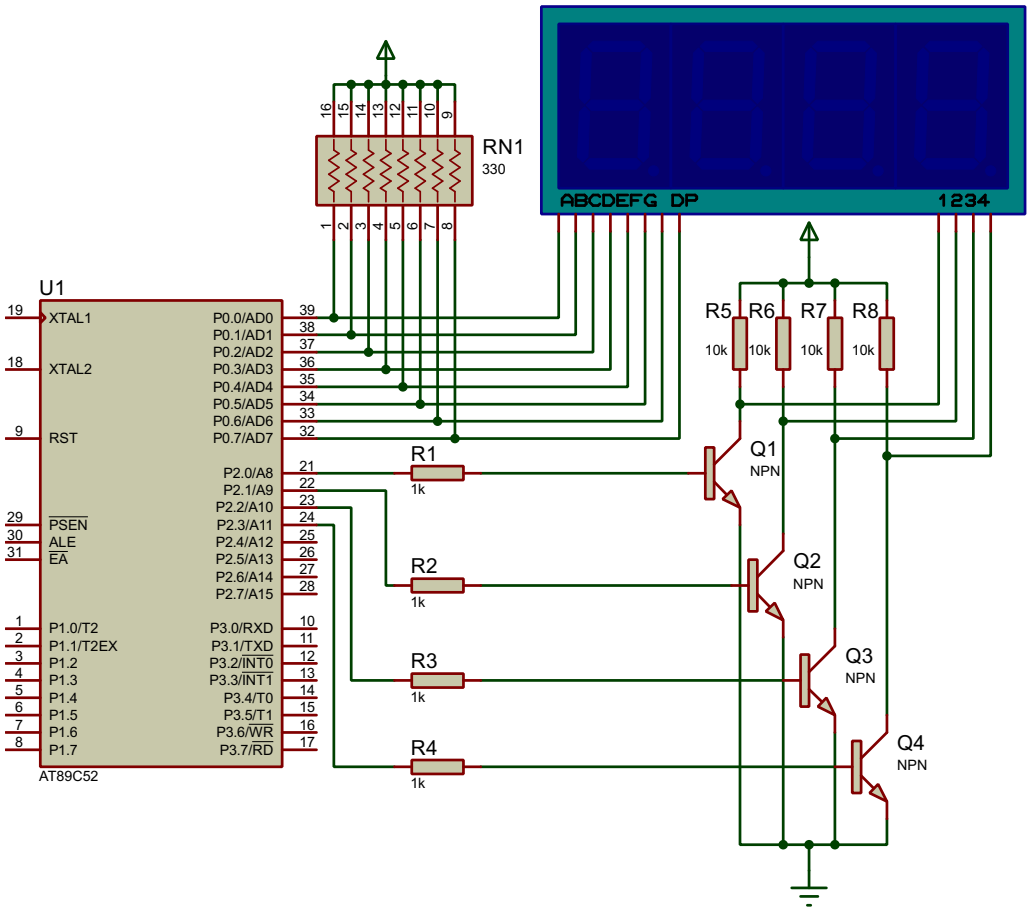


Figure 3: Circuit diagram for Proteus simulation

4 Lab Problems

Problem 1
Write a code to design a single digit decimal counter that counts up from 0 to 9 and back to 0. This process should repeat indefinitely.

Assembly Code

```
  ORG 00H
    ;HEX values for digits 0 to 9
    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
    ;display from 8 to 0
    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
```

Code 1: Problem 1 - Assembly

Embedded C Code

```
#include <reg51.h>
//HEX values for digits 0 to 9
unsigned char led_pattern[10] = {0x3f, 0x06,
     0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0
    x7f, 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 from 8 to 0
      display(i);
  }
}
```

Code 2: Problem 1 - Embedded C

Problem 2
Write a code to design a double digit decimal counter that counts up from 00 to 20 and back to 00 indefinitely.

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
```

Code 3: Problem 2 - Assembly

Embedded C Code

```
#include <reg51.h>
unsigned char led_pattern[10] = { 0x3f, 0x06
    , 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0
    x7f, 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, led[2];
  led[0] = i / 10;
  led[1] = i % 10;
  for(j=0; j<10; j++)  //Delay between two
    consecutive numbers
    for(i=0;i<2;i++)
    {
      P2 = 0x1 * (i + 1);
      P0 = led_pattern[led[i]];
      delay(40);//selected to avoid
    flickering
    }
}

void main(void)
{
  unsigned int i;
  while(1)
```

```
  {
    for(i=0; i<20; i++)
      display(i);
    for(i=20; i>0; i--)
      display(i);
  }
}
```

Code 4: Problem 2 - Embedded C

Problem 3

Write a code to display the first (N) numbers of the Fibonacci sequence, where the number (N) must be stored in a memory location and can be any integer from 1 to 10. The sequence should repeat indefinitely.

Assembly Code

```
;FIBONACCI SEQUENCE
    ORG 00H

    MOV P2,#00H
    MOV DPTR,#LABEL1
    MOV R0,#50H
    MOV R7,#9      ;Nummber of terms (N=9)
    MOV A,R7
    MOV R6,A

; Display 1st and 2nd numbers of 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

;Add consecutive terms to get next term
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 conversion and store in memory
    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

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


DISPLAY:MOVC A,@A+DPTR
    RET

;Lookup table
LABEL1: DB 3FH
    DB 06H
    DB 5BH
    DB 4FH
    DB 66H
    DB 6DH
    DB 7DH
    DB 07H
    DB 7FH
    DB 6FH

    END
```

Code 5: Problem 3 - Assembly

Embedded C Code

```
#include <reg51.h>
#define N 9
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++)//Delay between consecutive terms
  {
    P2 = 0x1;
    P0 = led_pattern[led1];
    delay(40);//selected to avoid flickering

    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]);
}
```

Code 6: Problem 3 - Embedded C

Problem 4
Write a code to generate the multiplication table of a number (N) stored in a memory location which can be any integer from 1 to 10. Repeat the sequence indefinitely.

Assembly Code

```
    ORG 00H

    MOV R7,#8 ;N=8
    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 conversion and store in memory
    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

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

DISPLAY:MOVC A,@A+DPTR
    RET

;Lookup table
LABEL1: DB 3FH
    DB 06H
    DB 5BH
    DB 4FH
    DB 66H
    DB 6DH
    DB 7DH
    DB 07H
    DB 7FH
    DB 6FH

    END
```

Code 7: Problem 4 - Assembly

Embedded C Code

```
#include <reg51.h>
#define N 8  //N=8
unsigned char led_pattern[10] = { 0x3f, 0x06
    , 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0
    x7f, 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);
}
```

Code 8: Problem 4 - Embedded C

Problem 5
Write a code to display your roll numbers in static format. Each student roll number should be of four characters. Display of student roll number should repeat indefinitely.

Assembly Code

```
    ORG 00H
;Digital drive pattern for E407
    MOV 40H,#79H
    MOV 41H,#66H
    MOV 42H,#3FH
    MOV 43H,#07H

REPEAT: MOV R0,#40H
    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

    AJMP REPEAT

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

    END
```

Code 9: Problem 5 - Assembly

Embedded C Code

```
#include <reg51.h>

unsigned char led_pattern[4] = {0x79, 0x66,
    0x3f, 0x07};


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

void display()
{
    P2 = 0x1;
    P0 = led_pattern[0];
    delay(10);//selected to avoid flickering
     and show as static

    P2 = 0x2;
    P0 = led_pattern[1];
    delay(10);

    P2 = 0x4;
    P0 = led_pattern[2];
    delay(10);

    P2 = 0x8;
    P0 = led_pattern[3];
    delay(10);
    }

void main(void)
{
  while(1)
      display();
}
```

Code 10: Problem 5 - Embedded C

Problem 6
Write a code to display your roll number in scrolling format, separated by using decimal point. Roll number should be scrolled towards the left and repeated indefinitely.

Assembly Code

```
    ORG 00H
    MOV 40H,#79H
    MOV 41H,#66H
    MOV 42H,#3FH
    MOV 43H,#07H
    MOV 44H,#79H
    MOV 45H,#66H
    MOV 46H,#3FH

REPEAT: MOV R0,#40H
    MOV R4,#04H
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
    ;Scrolling happens here
    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
```

Code 11: Problem 6 - Assembly

Embedded C Code

```
#include <reg51.h>

unsigned char scroll_pattern[7] = { 0x79, 0x66, 0x3f, 0x87,
                    0x79, 0x66, 0x3f};

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);//selected to avoid flickering

    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)
{
```

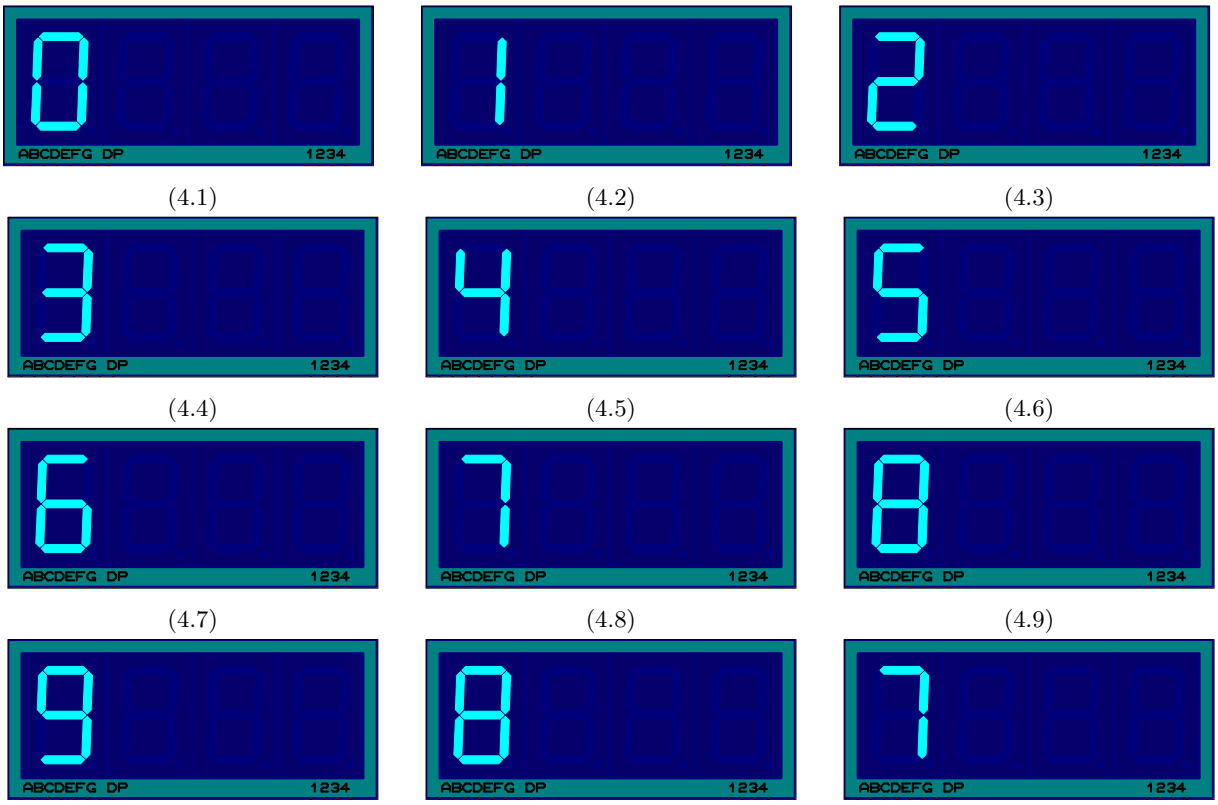
```
38   unsigned int i;
39   while(1)
40     for(i=4; i<8; i++)
41         display(i);//scrolling happens here
42 }
```

Code 12: Problem 6 - Embedded C

5 Observations

The observations for the 7-segment LED display are taken from Proteus VSM debugging. The different characters for a multiplexed display are actually displayed in quick succession with very less delay in order to trigger the persistence of vision of human eye, enabling us to use a single bus of segments to control multiplexed LED segments. The observations are taken based on a common cathode 7-segment LED display, but similar results can be obtained for a common anode type by using the look up table shown in Table 2.

Problem 1



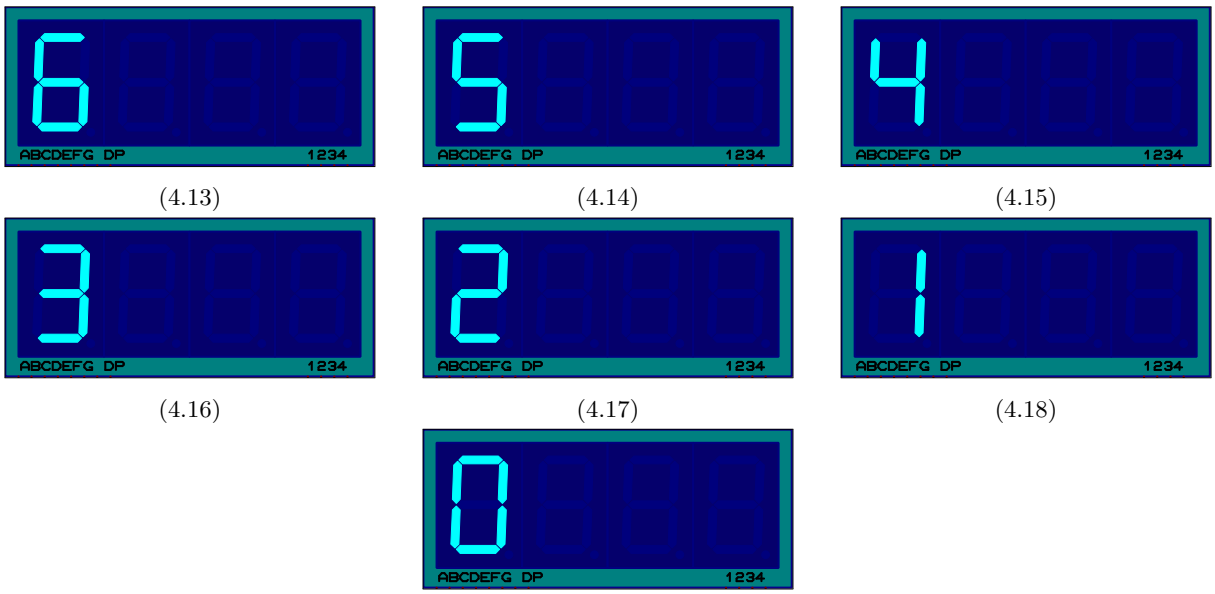
(4.1) (4.2) (4.3)

(4.4) (4.5) (4.6)

(4.7) (4.8) (4.9)

(4.10) (4.11) (4.12)

Figure 4: Observation for Problem 1



(4.13) (4.14) (4.15)

(4.16) (4.17) (4.18)

(4.19)

Figure 4: Observation for Problem 1 (continued)

Figure 4 shows the observations for Problem 1. The digits 0 to 9 are displayed on the first 7-segment LED display and then the display returns 8 to 0. That is to say, single digit decimal counter that counts up from 0 to 9 and back to 0 is observed. The loop continues indefinitely but only one cycle is presented in this report.

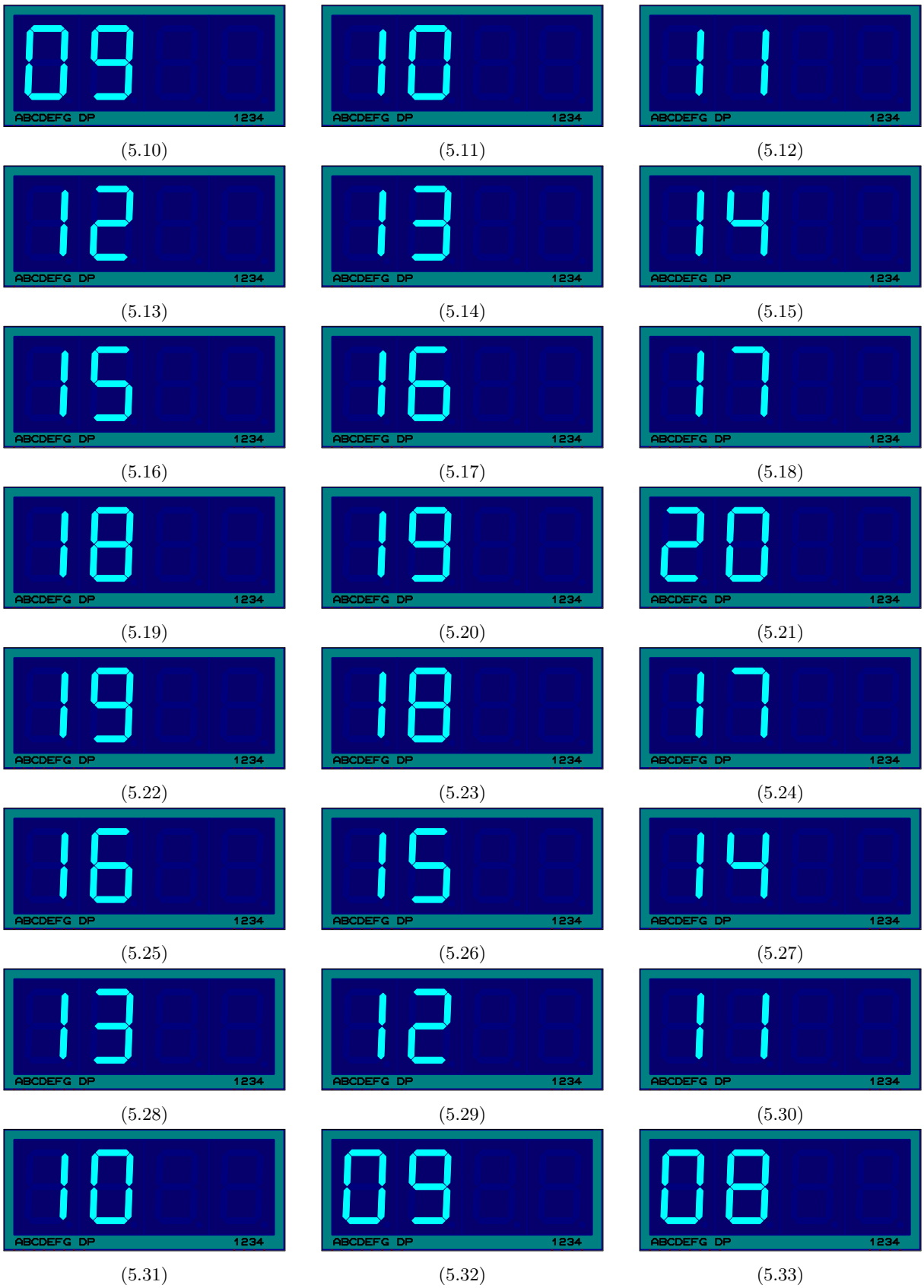
Problem 2

(5.1) (5.2) (5.3)

(5.4) (5.5) (5.6)

(5.7) (5.8) (5.9)

Figure 5: Observation for Problem 2



(5.10) (5.11) (5.12)

(5.13) (5.14) (5.15)

(5.16) (5.17) (5.18)

(5.19) (5.20) (5.21)

(5.22) (5.23) (5.24)

(5.25) (5.26) (5.27)

(5.28) (5.29) (5.30)

(5.31) (5.32) (5.33)

Figure 5: Observation for Problem 2 (continued)

(5.34) (5.35) (5.36)

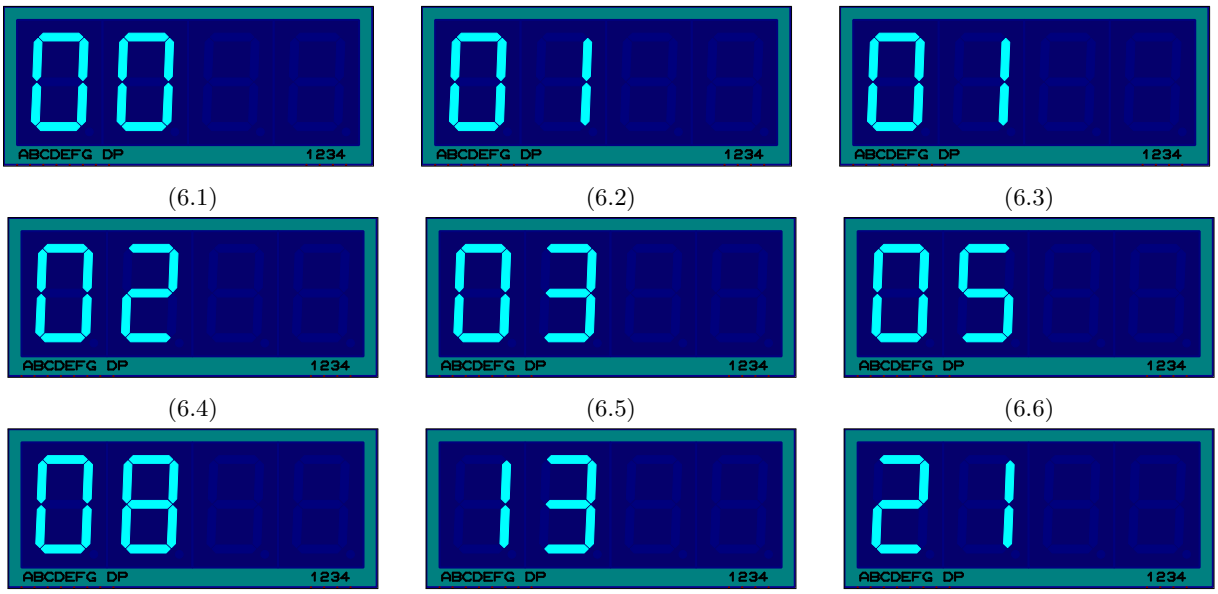
(5.37) (5.38) (5.39)

(5.40) (5.41)

Figure 5: Observation for Problem 2 (continued)

Figure 5 shows the observations for Problem 2. The digits 00 to 20 are displayed on the first and second 7-segment LED display and then the display returns 19 to 00. That is to say, double digit decimal counter that counts up from 00 to 20 and back to 00 is observed. The loop continues indefinitely but only one cycle is presented in this report.

Problem 3



(6.1) (6.2) (6.3)

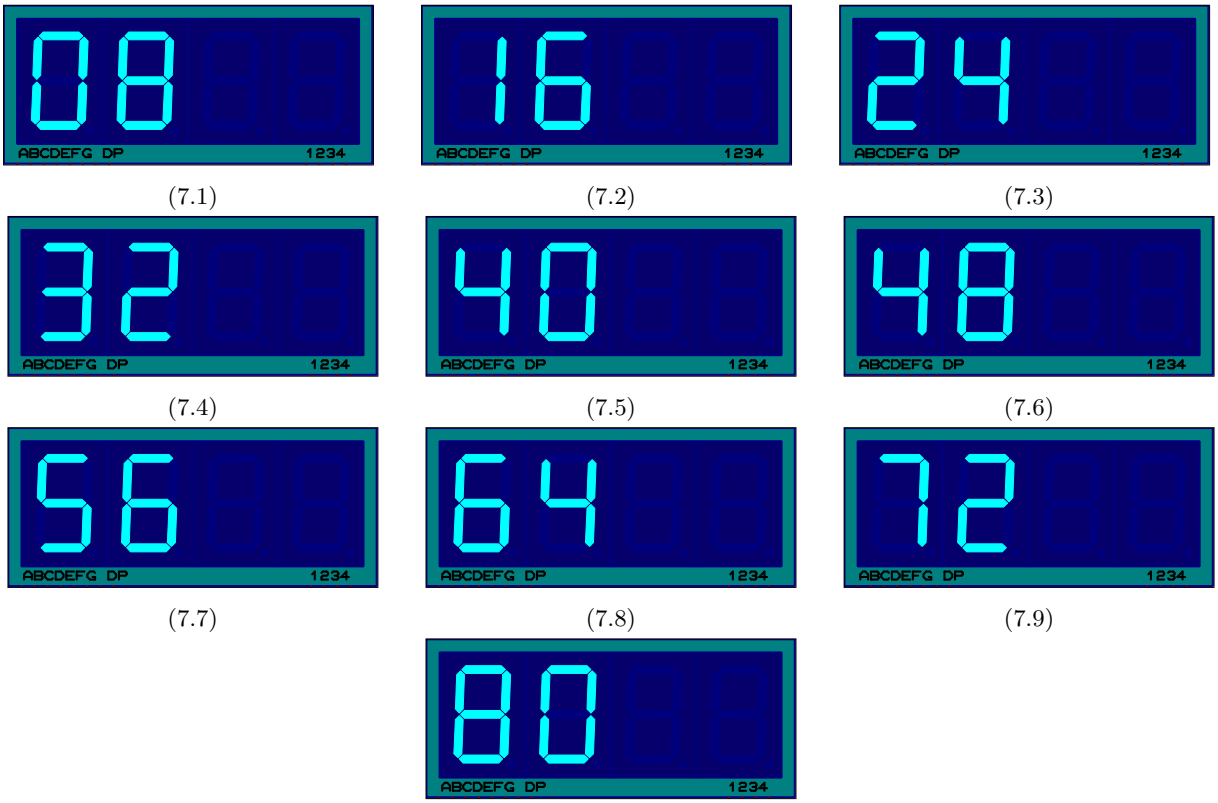
(6.4) (6.5) (6.6)

(6.7) (6.8) (6.9)

Figure 6: Observation for Problem 3

Figure 6 shows the observations for Problem 3. The fibonacci series up to N=9 terms is shown on the display. The display continually re-runs but only one cycle is included in this report.

Problem 4



(7.1) (7.2) (7.3)

(7.4) (7.5) (7.6)

(7.7) (7.8) (7.9)

(7.10)

Figure 7: Observation for Problem 4

Figure 7 shows the observations for Problem 4. The multiplication table for N=8 up to 10 terms is shown on the display. The multiplication table of any single digit number can be displayed by changing the value of N in the source codes. The display continually runs but only one cycle is included in this report.

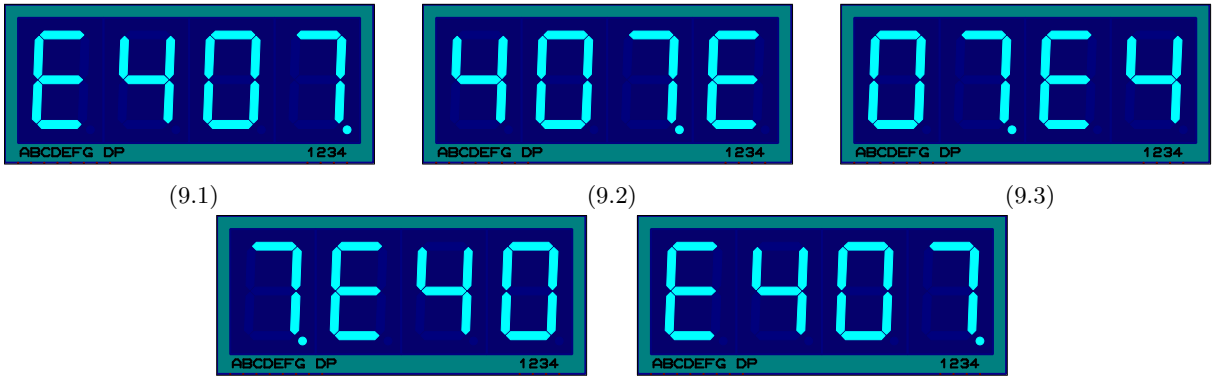
Problem 5

Figure 8: Observation for Problem 5

Figure 8 shows the observations for Problem 5. The roll number 407 preceded by a class code E (for BEX) is statically displayed on the 4 available 7-segment displays. The individual characters are actually shown with appropriate delay such that persistence of vision of human eye minimizes the flickering and hence a static

result is observed. Single step analysis of the simulation shows that the characters are displayed separately which is true since all the four 7-segment displays use a single bus for the input and hence work with latched select pins.

Problem 6



(9.1) (9.2) (9.3)

(9.4) (9.5)

Figure 9: Observation for Problem 6

Figure 8 shows the observations for Problem 6. The roll number 407 preceded by a class code E (for BEX) and terminated by a decimal point is displayed on the 7-segment LED display in a scrolling format. The pattern scrolls from right to left. The scrolling continues indefinitely but only one cycle is included in this report.

6 Discussion

In this lab experiment, interfaing 7-segment LED display in non-multiplication and multiplexed configurations were dealt with different levels of problems. Counters displaying single and double digits, fibonacci sequence, multiplication table allowed for segment manipulation of the 7-segment LED display based on the requirement. Static as well as scrolling content was also observed on the 7-segment display in multiplexed configuration. Varous programming concepts in assembly and embeddded C language for 8051/8052 microcontroller along with interfacing techniques with 7-segment displays were attained on the successful completion of the lab.

Bibliography

[1] R. Rathod et al. "Interfacing Multi-Digit 7-Segment with 8051 Microcontroller". en. In: *IJERT* (2020). [Online]. Available: URL: https://www.irjet.net/archives/V7/i3/IRJET-V7I3997.pdf.

Additional References

[2] H. Choudhary. "Seven segment multiplexing using 8051 microcontroller (AT89C51)- (Part 4/45)"". en. In: *Engineers Garage* (2020). [Online]. Available: URL: https://www.engineersgarage.com/8051-microcontroller/seven-segment-multiplexing-using-8051-microcontroller-at89c51-part-4-45/..

[3] "How to Count From 0 to 99 Using 8051 Microcontroller With 7 Segment Display"". en. In: *Instructables* (2020). [Online]. Available: URL: https://www.instructables.com/How-to-Count-From-0-to-99-Using-8051-Microcontroll/..

[4] Matthew Akanle and Victoria Oguntosin. "A digital indicator system with 7-segment display". In: *Journal of Physics: Conference Series* 1299 (Aug. 2019), p. 012139. DOI: 10.1088/1742-6596/1299/1/012139.