

### Write Program in KEIL Embedded C:

1. Write an 8051 C program to toggle all the bits of P0, P1, and P2 continuously with a 250 ms delay. Use the **sfr** keyword to declare the port addresses.

```
⇒ #include <REG51.h>
⇒
⇒ // Define SFR (Special Function Register) addresses for ports P0, P1, and
P2
⇒ sfr P0 = 0x80;
⇒ sfr P1 = 0x90;
⇒ sfr P2 = 0xA0;
⇒
⇒ void delay(unsigned int time_ms) {
⇒ unsigned int i, j;
⇒ for(i = 0; i < time_ms; i++) {
⇒ for(j = 0; j < 125; j++); // 1 ms delay approximately
⇒ }
⇒ }
⇒
⇒ void main() {
⇒ while(1) {
⇒ // Toggle all bits of port P0
⇒ P0 = ~P0;
⇒ // Toggle all bits of port P1
⇒ P1 = ~P1;
⇒ // Toggle all bits of port P2
⇒ P2 = ~P2;
⇒ // Delay for 250 ms
⇒ delay(250);
⇒ }
⇒ }
```

2. Write an 8051 C program to toggle all the bits of P0 and P2 continuously with a 250 ms delay. Using the inverting and Ex-OR operators, respectively.

```
⇒ #include <REG51.h>
⇒
⇒ void delay(unsigned int time_ms) {
⇒ unsigned int i, j;
⇒ for(i = 0; i < time_ms; i++) {
⇒ for(j = 0; j < 125; j++); // 1 ms delay approximately
⇒ }
⇒ }
⇒
⇒ void main() {
⇒ while(1) {
⇒ // Toggle all bits of port P0 using the inverting operator (~)
⇒ P0 = ~P0;
⇒
⇒ // Toggle all bits of port P2 using the XOR operator (^)
⇒ P2 = P2 ^ 0xFF; // 0xFF is the binary representation of all 1s
⇒
⇒ // Delay for 250 ms
⇒ delay(250);
⇒ }
⇒ }
```

### **To be done using EdSim51 simulator in 8051:**

1. Write an assembly program that displays the binary pattern from 0 to 255 (and back to 0) on the LEDs interfaced with port 1.

```
⇒ ORG 0x0000
⇒
⇒ MOV P1, #0x00
⇒
⇒ LOOP_UP:
⇒ MOV A, P1
⇒ INC A
⇒ MOV P1, A
⇒
⇒ ACALL DELAY
⇒ CJNE A, #0xFF, LOOP_UP
⇒
⇒ LOOP_DOWN:
```

- ⇒ MOV A, P1
- ⇒ DEC A
- ⇒ MOV P1, A
- ⇒ ACALL DELAY
- ⇒ JNZ LOOP\_DOWN
- ⇒ SJMP LOOP\_UP
- ⇒ DELAY:
- ⇒ MOV R2, #0xFF
- ⇒ DELAY\_LOOP:
- ⇒ NOP
- ⇒ NOP
- ⇒ NOP
- ⇒ DJNZ R2, DELAY\_LOOP
- ⇒ RET
- ⇒ END

2. Write an assembly language program that multiplexes the number 1234 on the four 7-segment displays.

- ⇒ ORG 0x0000
- ⇒ MOV P1, #0xFF
- ⇒ MAIN\_LOOP:
- ⇒ MOV A, #0x01
- ⇒ MOV P0, #0x7D

⇒ MOV P1, A

⇒ ACALL DELAY

⇒ MOV A, #0x02

⇒ MOV P0, #0xFB

⇒ MOV P1, A

⇒ ACALL DELAY

⇒ MOV A, #0x03

⇒ MOV P0, #0xFD

⇒ MOV P1, A

⇒ ACALL DELAY

⇒ MOV A, #0x04

⇒ MOV P0, #0xFE

⇒ MOV P1, A

⇒ ACALL DELAY

⇒ SJMP MAIN\_LOOP

⇒ DELAY:

⇒ MOV R2, #0xFF

⇒ DELAY\_LOOP:

⇒ NOP

⇒ NOP

⇒ NOP

⇒ DJNZ R2, DELAY\_LOOP

⇒ RET

⇒ END

3. Write a program to display message on the LCD of 8051 microcontroller

⇒ MOV 30H, #'M'

⇒ MOV 31H, #'N'

⇒ MOV 32H, #'N'

⇒ MOV 33H, #'I'

⇒ MOV 34H, #'T'

⇒ MOV 35H, #' '

⇒ MOV 36H, #'P'

⇒ MOV 37H, #'R'

⇒ MOV 38H, #'A'

⇒ MOV 39H, #'Y'

⇒ MOV 3AH, #'A'

⇒ MOV 3BH, #'G'

⇒ MOV 3CH, #'R'

⇒ MOV 3DH, #'A'

⇒ MOV 3EH, #'J'

⇒ CLR P1.3

⇒ CLR P1.7

⇒ CLR P1.6

⇒ SETB P1.5

⇒ CLR P1.4

- ⇒ SETB P1.2
- ⇒ CLR P1.2
- ⇒ CALL delay
- ⇒ SETB P1.2
- ⇒ CLR P1.2
- ⇒ SETB P1.7
- ⇒ SETB P1.2
- ⇒ CLR P1.2
- ⇒ CALL delay
- ⇒ CLR P1.7
- ⇒ CLR P1.6
- ⇒ CLR P1.5
- ⇒ CLR P1.4
- ⇒ SETB P1.2
- ⇒ CLR P1.2
- ⇒ SETB P1.6
- ⇒ SETB P1.5
- ⇒ SETB P1.2
- ⇒ CLR P1.2
- ⇒ CALL delay
- ⇒ CLR P1.7
- ⇒ CLR P1.6
- ⇒ CLR P1.5
- ⇒ CLR P1.4
- ⇒ SETB P1.2
- ⇒ CLR P1.2
- ⇒ SETB P1.7
- ⇒ SETB P1.6
- ⇒ SETB P1.5

```

⇒ SETB P1.4
⇒ SETB P1.2
⇒ CLR P1.2
⇒ CALL delay
⇒ SETB P1.3
⇒ MOV R1,#30H
⇒ loop: MOV A,@R1
⇒ JZ finish
⇒ CALL sendCharacter
⇒ INC R1
⇒ JMP loop
⇒ finish: JMP $
⇒ sendCharacter: MOV C, ACC.7
⇒ MOV P1.7, C
⇒ MOV C, ACC.6
⇒ MOV P1.6, C
⇒ MOV C, ACC.5
⇒ MOV P1.5, C
⇒ MOV C, ACC.4
⇒ MOV P1.4, C
⇒ SETB P1.2
⇒ CLR P1.2
⇒ MOV C, ACC.3
⇒ MOV P1.7, C
⇒ MOV C, ACC.2
⇒ MOV P1.6, C
⇒ MOV C, ACC.1
⇒ MOV P1.5, C
⇒ MOV C, ACC.0
⇒ MOV P1.4, C
⇒ SETB P1.2
⇒ CLR P1.2
⇒ CALL delay
⇒ delay: MOV R0, #50
⇒ DJNZ R0, $
⇒ RET

```

4. Write a program to display your name on the LCD of 8051 microcontroller

```

⇒ MOV 30H, #'D'

```

⇒ **MOV 31H, #'I'**

⇒ **MOV 32H, #'N'**

⇒ **MOV 33H, #'G'**

⇒ **MOV 34H, #'L'**

⇒ **MOV 35H, #'O'**

⇒ **MOV 36H, #' '**

⇒ **MOV 37H, #'K'**

⇒ **MOV 38H, #'O'**

⇒ **MOV 39H, #'D'**

⇒ **MOV 3AH, #'A'**

⇒ **MOV 3BH, #'K'**

⇒ **CLR P1.3**

⇒ **CLR P1.7**

⇒ **CLR P1.6**

⇒ **SETB P1.5**

⇒ **CLR P1.4**

⇒ **SETB P1.2**

⇒ **CLR P1.2**

⇒ **CALL delay**

⇒ **SETB P1.2**

⇒ **CLR P1.2**

⇒ **SETB P1.7**

⇒ **SETB P1.2**

⇒ **CLR P1.2**

⇒ **CALL delay**



- ⇒ **CLR P1.7**
- ⇒ **CLR P1.6**
- ⇒ **CLR P1.5**
- ⇒ **CLR P1.4**
- ⇒ **SETB P1.2**
- ⇒ **CLR P1.2**
- ⇒ **SETB P1.6**
- ⇒ **SETB P1.5**
- ⇒ **SETB P1.2**
- ⇒ **CLR P1.2**
- ⇒ **CALL delay**
- ⇒ **CLR P1.7**
- ⇒ **CLR P1.6**
- ⇒ **CLr P1.5**
- ⇒ **CLR P1.4**
- ⇒ **SETB P1.2**
- ⇒ **CLR P1.2**
- ⇒ **SETB P1.7**
- ⇒ **SETB P1.6**
- ⇒ **SETB P1.5**
- ⇒ **SETB P1.4**
- ⇒ **SETB P1.2**
- ⇒ **CLR P1.2**
- ⇒ **CALL delay**
- ⇒ **SETB P1.3**
- ⇒ **MOV R1,#30H**
- ⇒ **loop: MOV A,@R1**
- ⇒ **JZ finish**
- ⇒ **CALL sendCharacter**
- ⇒ **INC R1**
- ⇒ **JMP loop**
- ⇒ **finish: JMP \$**

⇒ **sendCharacter: MOV C, ACC.7**

⇒ **MOV P1.7, C**

⇒ **MOV C, ACC.6**

⇒ **MOV P1.6, C**

⇒ **MOV C, ACC.5**

⇒ **MOV P1.5, C**

⇒ **MOV C, ACC.4**

⇒ **MOV P1.4, C**

⇒ **SETB P1.2**

⇒ **CLR P1.2**

⇒ **MOV C, ACC.3**

⇒ **MOV P1.7, C**

⇒ **MOV C, ACC.2**

⇒ **MOV P1.6, C**

⇒ **MOV C, ACC.1**

⇒ **MOV P1.5, C**

⇒ **MOV C, ACC.0**

⇒ **MOV P1.4, C**

⇒

⇒ **SETB P1.2**

⇒ **CLR P1.2**

⇒

⇒ **CALL delay**

⇒

⇒ **delay: MOV R0, #50**

⇒ **DJNZ R0, \$**

⇒ **RET**