**Embedded Systems Lab**

**Assignment 7**

**Name** – Neelangshu Roy

**Reg No** – 20214060

**Group** – C1

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.

Code –

#include<reg51.h>

void createDelay(int t)

{

int i,j;

t\*=10;

for(i=0;i<t;i++)

for(j=0;j<1275;j++);

}

void main(void)

{

unsigned int n,k;

int turn=0;

P0=0x6c;

k=255-P0;

P1=0x97;

P2=0x33;

P3=0xff;

while(1)

{

if(!turn)P0=k;

else P0=255-k;

turn^=1;

n=P1;

n=255-n;

P1=n;

n=P2;

n=255-n;

P2=n;

n=P3;

n=255-n;

P3=n;

createDelay(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.

Code –

#include<reg51.h>

void createDelay(int t)

{

int i,j;

t\*=10;

for(i=0;i<t;i++)

for(j=0;j<1275;j++);

}

void main(void)

{

unsigned int k;

int turn=0,i;

int bits[]={0,0,0,0,0,0,0,0};

P0=0x6c;

k=255-P0;

P1=0x97;

P2=0x33;

P3=0xff;

for(i=0;i<8;i++)

{

bits[i]=P2&1;

P2>>=1;

}

while(1)

{

if(!turn)P0=k;

else P0=255-k;

turn^=1;

P1^=255;

for(i=7;i>=0;i--)

{

P2+=bits[i];

bits[i]=!bits[i];

P2<<=1;

}

P3=~P3;

createDelay(250);

}

}

To be done using EdSim51 simulator in 8051:

Q1. Write an assembly program that displays the binary pattern from 0 to 255 (and back to 0) on the LEDs interfaced with port 1.

Code –

orl p0, #80h

mov p1, #0

loop: inc p1

jmp loop

end

Q2. Write an assembly language program that multiplexes the number 1234 on the four 7-segment displays.

Code –

orl p0, #80h

loop: nop

mov p1, #0ffh

anl p3, #0e7h

mov p1, #99h

nop

mov p1, #0ffh

setb p3.3

mov p1, #0b0h

nop

mov p1, #0ffh

anl p3, #0f7h

setb p3.4

mov p1, #0a4h

nop

mov p1, #0ffh

setb p3.3

mov p1, #0f9h

jmp loop

end

Q3. Write a program to display message on the LCD of 8051 microcontroller.

Code –

; put data in RAM

MOV 30H, #'T'

MOV 31H, #'H'

MOV 32H, #'I'

MOV 33H, #'S'

MOV 34H, #' '

MOV 35H, #'I'

MOV 36H, #'S'

MOV 37H, #' '

MOV 38H, #'A'

MOV 39H, #' '

MOV 3AH, #'T'

MOV 3BH, #'E'

MOV 3CH, #'X'

MOV 3DH, #'T'

; initialise the display

; see instruction set for details

CLR P1.3 ; clear RS - indicates that instructions are being sent to the module

; function set

CLR P1.7 ; |

CLR P1.6 ; |

SETB P1.5 ; |

CLR P1.4 ; | high nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

CALL delay ; wait for BF to clear

; function set sent for first time - tells module to go into 4-bit mode

; Why is function set high nibble sent twice? See 4-bit operation on pages 39 and 42 of HD44780.pdf.

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

; same function set high nibble sent a second time

SETB P1.7 ; low nibble set (only P1.7 needed to be changed)

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

; function set low nibble sent

CALL delay ; wait for BF to clear

; entry mode set

; set to increment with no shift

CLR P1.7 ; |

CLR P1.6 ; |

CLR P1.5 ; |

CLR P1.4 ; | high nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

SETB P1.6 ; |

SETB P1.5 ; |low nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

CALL delay ; wait for BF to clear

; display on/off control

; the display is turned on, the cursor is turned on and blinking is turned on

CLR P1.7 ; |

CLR P1.6 ; |

CLR P1.5 ; |

CLR P1.4 ; | high nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

SETB P1.7 ; |

SETB P1.6 ; |

SETB P1.5 ; |

SETB P1.4 ; | low nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

CALL delay ; wait for BF to clear

; send data

SETB P1.3 ; clear RS - indicates that data is being sent to module

MOV R1, #30H ; data to be sent to LCD is stored in 8051 RAM, starting at location 30H

loop:

MOV A, @R1 ; move data pointed to by R1 to A

JZ finish ; if A is 0, then end of data has been reached - jump out of loop

CALL sendCharacter ; send data in A to LCD module

INC R1 ; point to next piece of data

JMP loop ; repeat

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 ; | high nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

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 ; | low nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

CALL delay ; wait for BF to clear

delay:

MOV R0, #50

DJNZ R0, $

RET

Q4. Write a program to display your name on the LCD of 8051 microcontroller.

Code –

; put data in RAM

MOV 30H, #'N'

MOV 31H, #'E'

MOV 32H, #'E'

MOV 33H, #'L'

MOV 34H, #'A'

MOV 35H, #'N'

MOV 36H, #'G'

MOV 37H, #'S'

MOV 38H, #'H'

MOV 39H, #'U'

MOV 3AH, #' '

MOV 3BH, #'R'

MOV 3CH, #'O'

MOV 3DH, #'Y'

; initialise the display

; see instruction set for details

CLR P1.3 ; clear RS - indicates that instructions are being sent to the module

; function set

CLR P1.7 ; |

CLR P1.6 ; |

SETB P1.5 ; |

CLR P1.4 ; | high nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

CALL delay ; wait for BF to clear

; function set sent for first time - tells module to go into 4-bit mode

; Why is function set high nibble sent twice? See 4-bit operation on pages 39 and 42 of HD44780.pdf.

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

; same function set high nibble sent a second time

SETB P1.7 ; low nibble set (only P1.7 needed to be changed)

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

; function set low nibble sent

CALL delay ; wait for BF to clear

; entry mode set

; set to increment with no shift

CLR P1.7 ; |

CLR P1.6 ; |

CLR P1.5 ; |

CLR P1.4 ; | high nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

SETB P1.6 ; |

SETB P1.5 ; |low nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

CALL delay ; wait for BF to clear

; display on/off control

; the display is turned on, the cursor is turned on and blinking is turned on

CLR P1.7 ; |

CLR P1.6 ; |

CLR P1.5 ; |

CLR P1.4 ; | high nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

SETB P1.7 ; |

SETB P1.6 ; |

SETB P1.5 ; |

SETB P1.4 ; | low nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

CALL delay ; wait for BF to clear

; send data

SETB P1.3 ; clear RS - indicates that data is being sent to module

MOV R1, #30H ; data to be sent to LCD is stored in 8051 RAM, starting at location 30H

loop:

MOV A, @R1 ; move data pointed to by R1 to A

JZ finish ; if A is 0, then end of data has been reached - jump out of loop

CALL sendCharacter ; send data in A to LCD module

INC R1 ; point to next piece of data

JMP loop ; repeat

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 ; | high nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

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 ; | low nibble set

SETB P1.2 ; |

CLR P1.2 ; | negative edge on E

CALL delay ; wait for BF to clear

delay:

MOV R0, #50

DJNZ R0, $

RET