Bahria University (Karachi Campus)

Department of Software Engineering

**Assignment # 2 – Fall 2023**

|  |  |  |
| --- | --- | --- |
| Course Title: | **Embedded Systems** | Course Code: **CEN-439** |
| Class: | **BSE-5(B)** | Shift: **Morning** |
| Name: | **Muhammad Shoaib Akhter Qadri** | Reg No: **79290** |
| Course Instructor: | **Dr. Qamaruddin Memon** | Enroll No; **02-131212-009** |

# Draw the Block diagram of seven segment display on Multisim/Proteus and write its code in C and Assembly

## CODE:

### #include <htc.h> #define Port\_0 P0

unsigned char displayPatterns[] = {

0x40, // 0

0x79, // 1

0x36, // 2

0x48, // 3

0x19, // 4

0x12, // 5

0x02, // 6

0x78, // 7

0x00, // 8

0x10 // 9

}

void delay(unsigned int milliseconds) {

unsigned int i, j;

for (i = 0; i < milliseconds; i++) {

for (j = 0; j < 110; j++);

}

}

void main() {

unsigned char i;

while (1) {

for (i = 0; i < 10; i++) {

Port\_0 = displayPatterns[i]; // Display the current digit delay(10); // Delay for a while to display the digit

}

}

}

**Assembly Code:**

$MOD51; This includes 8051 definitions for the Metalink assembler org 0000h

### repeat:

mov p0,#10000001b ; displaying 0 acall delay

mov p0,#11001111b ; displaying 1 acall delay

mov p0,#10010010b ; displaying 2 acall delay

mov p0,#10000110b ; displaying 3 acall delay

mov p0,#11001100b ; displaying 4 acall delay

mov p0,#10100100b ; displaying 5 acall delay

mov p0,#10100000b ; displaying 6 acall delay

mov p0,#10001111b ; displaying 7 acall delay

mov p0,#10000000b ; displaying 8 acall delay

mov p0,#10000100b ; displaying 9 acall delay

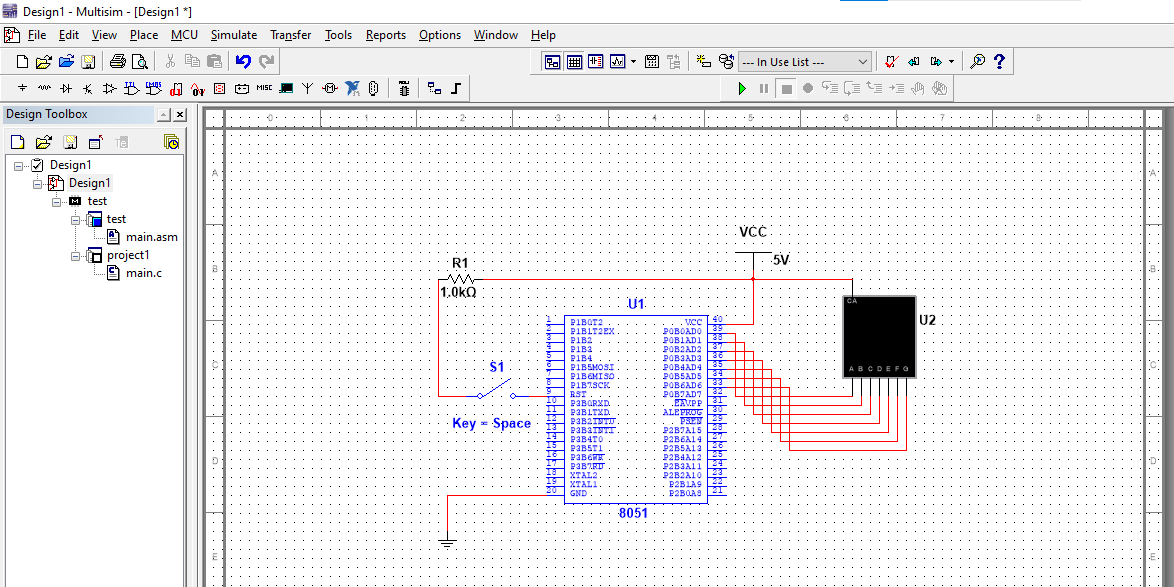
sjmp repeat

### delay:

mov r3,#010h l3:mov r2,#0ffh l2:mov r1,#0ffh l1:djnz r1,l1 djnz r2,l2

djnz r3,l3 ret

END



**Output:**

# Draw the Block diagram of seven segment display on Multisim/Proteous and write its code in C and Assembly Write a program in Assembly and C to display a digit from 0 to 9 in both seven segment display.

## C CODE

### #include <htc.h> #define Port\_0 P0 #define Port\_2 P2

displayPatterns[] = {

0x40, // 0

0x79, // 1

0x36, // 2

0x48, // 3

0x19, // 4

0x12, // 5

0x02, // 6

0x78, // 7

0x00, // 8

0x10 // 9

};

void delay(unsigned int milliseconds) { unsigned int

i, j;

for (i = 0; i < milliseconds; i++) { for (j = 0; j < 110; j++);

}

}

void main() {

unsigned char i; while (1) {

for (i = 0; i < 10; i++) {

Port\_0 = displayPatterns[i]; // Display the current digit Port\_2 = displayPatterns[i]; // Display the current digit

delay(10);

}

}}

**Assembly Code:**

$MOD51; This includes 8051 definitions for the Metalink assembler org 0000h

|  |  |  |  |
| --- | --- | --- | --- |
| **repeat:**  mov p0,#10000001b | ; | displaying | 0 |
| mov p2,#10000001b | ; | displaying | 0 |
| acall delay  mov p0,#11001111b | ; | displaying | 1 |
| mov p2,#11001111b | ; | displaying | 1 |
| acall delay  mov p0,#10010010b | ; | displaying | 2 |
| mov p2,#10010010b | ; | displaying | 2 |
| acall delay  mov p0,#10000110b | ; | displaying | 3 |
| mov p2,#10000110b | ; | displaying | 3 |
| acall delay  mov p0,#11001100b | ; | displaying | 4 |
| mov p2,#11001100b | ; | displaying | 4 |
| acall delay  mov p0,#10100100b | ; | displaying | 5 |
| mov p2,#10100100b | ; | displaying | 5 |
| acall delay  mov p0,#10100000b | ; | displaying | 6 |
| mov p2,#10100000b | ; | displaying | 6 |
| acall delay  mov p0,#10001111b | ; | displaying | 7 |
| mov p2,#10001111b | ; | displaying | 7 |
| acall delay  mov p0,#10000000b | ; | displaying | 8 |
| mov p2,#10000000b | ; | displaying | 8 |
| acall delay  mov p0,#10000100b | ; | displaying | 9 |
| mov p2,#10000100b acall delay  sjmp repeat | ; | displaying | 9 |
| **delay:**  mov r3,#010h l3:mov r2,#0ffh l2:mov r1,#0ffh l1:djnz r1,l1 djnz r2,l2  djnz r3,l3 ret  END  Output: |  |  |  |

# A screenshot of a computer Description automatically generatedWrite a program to display a welcome to bahria in LCD display using 8051 and Arduino.

## C CODE

### #include <htc.h> #include <stdio.h>

#define LCD\_data P2 #define LCD\_D7 P27 #define LCD\_rs P10 #define LCD\_rw P11 #define LCD\_en P12

void LCD\_init(); void LCD\_busy();

void LCD\_command(unsigned char ); void LCD\_senddata(unsigned char ); void LCD\_sendstring(unsigned char \*);

void main()

{

unsigned char msg[] = "WELCOME TO BAHRIA";

LCD\_init();

LCD\_command(0x80);

LCD\_sendstring(msg);

while(1);

}

void LCD\_init()

{

LCD\_data = 0x38;

LCD\_rw = 0;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LCD\_en  LCD\_en | | =  = | 1;  0; | //Enable H->L |
|  | LCD\_busy(); | |  | //Wait for LCD to process the command |
|  | LCD\_data = LCD\_rs =  LCD\_rw =  LCD\_en =  LCD\_en = LCD\_busy(); | | 0x0D;  0;  0;  1;  0; | //Display on, Curson blinking command  //Selected command register  //We are writing in data register  //Enable H->L  //Wait for LCD to process the command |
|  | LCD\_data = LCD\_rs =  LCD\_rw =  LCD\_en =  LCD\_en = LCD\_busy(); | | 0x01;  0;  0;  1;  0; | //Clear LCD  //Selected command register  //We are writing in data register  //Enable H->L  //Wait for LCD to process the command |
| } | LCD\_data = LCD\_rs =  LCD\_rw =  LCD\_en = LCD\_busy(); | | 0x06;  0;  0;  1; | //Entry mode, auto increment with no shift  //Selected command register  //We are writing in data register  //Enable H->L |

void LCD\_busy()

{

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LCD\_D7 | = 1; | //Make D7th bit of LCD as i/p | | |
| LCD\_en | = 1; | //Make port pin as o/p | | |
| LCD\_rs | = 0; | //Selected command register | | |
| LCD\_rw | = 1; | //We are reading | | |
| while(LCD\_D7)  {  LCD\_en | | = | 0; | //Enable H->L |
| LCD\_en | | = | 1; |  |

}

}

void LCD\_command(unsigned char var)

{

LCD\_data = var; //Function set: 2 Line, 8-bit, 5x7 dots LCD\_rs = 0; //Selected command register

LCD\_rw = 0; //We are writing in instruction register LCD\_en = 1; //Enable H->L

LCD\_en = 0;

LCD\_busy(); //Wait for LCD to process the command

}

void LCD\_senddata(unsigned char var)

{

LCD\_data = var; //Function set: 2 Line, 8-bit, 5x7 dots LCD\_rs = 1; //Selected data register

LCD\_rw = 0; //We are writing

LCD\_en = 1; //Enable H->L LCD\_en = 0;

LCD\_busy(); //Wait for LCD to process the command

}

void LCD\_sendstring(unsigned char \*var)

{

while(\*var) //till string ends LCD\_senddata(\*var++); //send characters one by one

|  |  |  |
| --- | --- | --- |
| } |  | |
| $MOD51 RS | EQU | P0.0 |
| RW | EQU | P0.1 |
| EN | EQU | P0.2 |
| DATA\_PORT | EQU | P2 |
| NBYTES | EQU | R0 |
| BYTE\_IDX | EQU | R1 |
| ORG 0H |  |  |

**Assembly Code:**

; Initialize LCD INIT:

MOV DPTR, #INIT\_CMND

MOV NBYTES, #3 ; INIT\_CMND has 3

commands = 3 bytes

ACALL SEND\_CMND\_BYTES ; Call

SEND\_CMND\_BYTES subroutine

; Main program MAIN:

SEND\_DATA1:

MOV DPTR, #LINE1 ; Load DPTR with

command to begin cursor at line 1

MOV NBYTES, #2 ; LINE1 has 2 commands =

2 bytes

ACALL SEND\_CMND\_BYTES ; Call

SEND\_CMND\_BYTES subroutine

MOV DPTR, #DATA1 ; Load DPTR

with data for line 1

MOV NBYTES, #5 ; DATA1 has 5 char = 5

bytes

ACALL SEND\_DATA\_BYTES ; Call

SEND\_DATA\_BYTES subroutine

SEND\_DATA2:

MOV DPTR, #LINE2 ; Load DPTR with

command to begin cursor at line 1

MOV NBYTES, #1 ; LINE2 has 1 command =

1 byte

ACALL SEND\_CMND\_BYTES ; Call

SEND\_CMND\_BYTES subroutine

MOV DPTR, #DATA2 ; Load DPTR

with data for line 1

MOV NBYTES, #5 ; DATA2 has 5 char = 5

byte

ACALL SEND\_DATA\_BYTES ; Call

SEND\_DATA\_BYTES subroutine

SJMP MAIN ; Jump back to MAIN (repeat main program)

; SEND\_DATA\_BYTES subroutine: write one byte of data to the LCD at a time

SEND\_DATA\_BYTES:

MOV BYTE\_IDX, #0 ; Initialize byte idx at 0

SEND\_DATA\_BYTE:

MOV A, BYTE\_IDX ; Load A with the value of BYTE\_IDX

ACALL WRT\_DATA ; Send data to LCD

ACALL DELAY ; Call DELAY

subroutine

INC BYTE\_IDX ; Increment byte

idx

DJNZ NBYTES, SEND\_DATA\_BYTE ; Repeat for

each byte of data (loop until NBYTES = 0) RET

; SEND\_CMND\_BYTES subroutine: write one byte of command(s) to the LCD at a time SEND\_CMND\_BYTES:

MOV BYTE\_IDX, #0 ; Initialize byte idx at 0

SEND\_CMND\_BYTE:

MOV A, BYTE\_IDX ; Load A with the value of BYTE\_IDX

ACALL WRT\_CMND ; Send command to

LCD

ACALL DELAY ; Call DELAY

subroutine

INC BYTE\_IDX ; Increment byte

idx

DJNZ NBYTES, SEND\_CMND\_BYTE ; Repeat for

each byte of command (loop until NBYTES = 0) RET

; WRT\_CMND subroutine: send command to LCD WRT\_CMND:

MOVC A, @A+DPTR ; Address of the desired byte in code space is formed by adding A + DPTR

MOV DATA\_PORT, A ; Load DATA\_PORT with

contents of A

CLR RS ; RS = 0 for command

CLR RW ; RW = 0 for write

SETB EN ; EN = 1 for high pulse

ACALL DELAY ; Call DELAY subroutine

CLR EN ; EN = 0 for low pulse RET

; WRT\_DATA subroutine: send data to LCD and display WRT\_DATA:

MOVC A, @A+DPTR ; Address of the desired byte in code space is formed by adding A + DPTR

MOV DATA\_PORT, A ; Load DATA\_PORT with

contents of A

SETB RS ; RS = 1 for data

CLR RW ; RW = 0 for write

SETB EN ; EN = 1 for high pulse

ACALL DELAY ; Call DELAY subroutine

CLR EN ; EN = 0 for low pulse RET

; DELAY subroutine

DELAY: MOV R3, #255 ; Load R3 with 255

L2: MOV R4, #2 ; Load R4 with 2

L1: DJNZ R4, L1 ; Decrement R4, if not zero repeat L1

DJNZ R3, L2 ; Decrement R3, if not zero repeat L1

RET

; Define commands to initialize LCD display

; 38H: 8-bit, 2 line, 5x7 dots

; 0EH: Display ON cursor, ON

; 06H: Auto increment mode, i.e., when we send char, cursor position moves right

INIT\_CMND: DB 38H, 0EH, 06H

; Define data to display on lines 1 and 2 of LCD DATA1: DB Welcome To

DATA2: DB 'Bahria'

; Define commands to go to line 1 of display

; 01H: Clear display

; 80H: Bring cursor to line 1 LINE1: DB 01H, 80H

; Define command to go to line 2 of display

; 0C0H: Bring cursor to line 2 LINE2: DB 0C0H

END