**Lab:5**

**Interfacing Seven Segment Display to 8051 Development kit**



**MBSD Lab**

**Spring 2023**

**Submitted by:**

**Muhammad Fahad Khan [20 Pwcse1940]**

**Semester: 6**

“On my honor, as a student of University of Engineering and Technology Peshawar, I have neither nor received unauthorized assistance on this academic work”

**Submitted to:**

**Dr:Amad Khalil**

1. **Display 0-9 on Seven Segment Display.**

**Answer:**

To display numbers 0-9 on a seven-segment display using an 8051 microcontroller, you need to connect the appropriate GPIO pins of the microcontroller to the segments of the display. Each segment of the seven-segment display should be connected to a separate GPIO pin. Here's an example code that demonstrates how to display numbers 0-9 on a common cathode seven-segment display using an 8051 microcontroller:

**Source Code:**

#include <reg51.h>

// Define the GPIO pins for seven-segment display segments

sbit segmentA = P2^0;

sbit segmentB = P2^1;

sbit segmentC = P2^2;

sbit segmentD = P2^3;

sbit segmentE = P2^4;

sbit segmentF = P2^5;

sbit segmentG = P2^6;

// Define the GPIO pins for the common cathodes of the display

sbit digit1 = P1^0;

sbit digit2 = P1^1;

sbit digit3 = P1^2;

sbit digit4 = P1^3;

// Define the patterns for each digit from 0 to 9

unsigned char digitPatterns[] = {

0xC0, // 0

0xF9, // 1

0xA4, // 2

0xB0, // 3

0x99, // 4

0x92, // 5

0x82, // 6

0xF8, // 7

0x80, // 8

0x90 // 9

};

void delay(unsigned int time) {

unsigned int i, j;

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

for (j = 0; j < 1275; j++); // Adjust this value for the desired delay

}

void displayDigit(unsigned char digit) {

segmentA = digitPatterns[digit] & 0x01;

segmentB = digitPatterns[digit] & 0x02;

segmentC = digitPatterns[digit] & 0x04;

segmentD = digitPatterns[digit] & 0x08;

segmentE = digitPatterns[digit] & 0x10;

segmentF = digitPatterns[digit] & 0x20;

segmentG = digitPatterns[digit] & 0x40;

}

void main() {

while (1) {

unsigned char i;

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

displayDigit(i);

// Display the digit for a certain time

digit1 = 0;

digit2 = 1;

digit3 = 1;

digit4 = 1;

delay(500); // Delay for 0.5 seconds

// Turn off all digits

digit1 = 1;

digit2 = 1;

digit3 = 1;

digit4 = 1;

delay(200); // Delay between digits

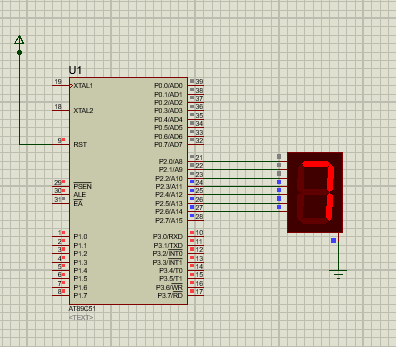
}

}

}

```

**Schematric Diagram:**

****

1. **Write a program to count up to 00-99 using seven segment display using with single port**

**Source Code:**

org 000h

start:

    MOV A,#0

    mov R1,#10

TEN:

    mov R2,#10

UNIT1:

    MOV P2,A

    ADD A,#1

    acall DELAY

    DJNZ R2,UNIT1

    ADD A,#6     ;After each 10 digit add 6 to convert hex to decimal

    DJNZ R1,TEN

LJMP start

DELAY:

 MOV R4,#255

AGAIN:

 MOV R5,#255

HERE:

 NOP

 NOP

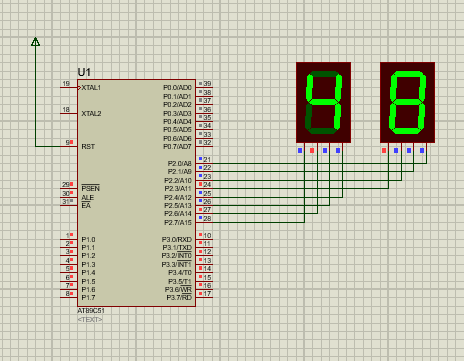
 DJNZ R5,HERE

 DJNZ R4,AGAIN

 RET

END

**Schematric Diagram:**

****

1. **Write a program to count up to 0000-9999 using four seven segment display with single port**

To count up to 0000-9999 using four seven-segment displays connected to a single port of an 8051 microcontroller, you can utilize multiplexing technique. Here's an example code that demonstrates how to achieve this:

**Source Code:**

#include <reg51.h>

// Define the GPIO pins for the seven-segment displays

sbit segmentA = P1^0;

sbit segmentB = P1^1;

sbit segmentC = P1^2;

sbit segmentD = P1^3;

sbit segmentE = P1^4;

sbit segmentF = P1^5;

sbit segmentG = P1^6;

sbit segmentDP = P1^7;

// Define the GPIO pins for the common cathodes of the displays

sbit digit1 = P2^0;

sbit digit2 = P2^1;

sbit digit3 = P2^2;

sbit digit4 = P2^3;

// Define the patterns for each digit from 0 to 9

unsigned char digitPatterns[] = {

0xC0, // 0

0xF9, // 1

0xA4, // 2

0xB0, // 3

0x99, // 4

0x92, // 5

0x82, // 6

0xF8, // 7

0x80, // 8

0x90 // 9

};

void delay(unsigned int time) {

unsigned int i, j;

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

for (j = 0; j < 1275; j++); // Adjust this value for the desired delay

}

void displayDigit(unsigned char digit, unsigned char value) {

// Turn off all digits

digit1 = 1;

digit2 = 1;

digit3 = 1;

digit4 = 1;

// Set segment patterns for the selected digit

segmentA = digitPatterns[value] & 0x01;

segmentB = digitPatterns[value] & 0x02;

segmentC = digitPatterns[value] & 0x04;

segmentD = digitPatterns[value] & 0x08;

segmentE = digitPatterns[value] & 0x10;

segmentF = digitPatterns[value] & 0x20;

segmentG = digitPatterns[value] & 0x40;

segmentDP = digitPatterns[value] & 0x80;

// Enable the selected digit

switch (digit) {

case 1:

digit1 = 0;

break;

case 2:

digit2 = 0;

break;

case 3:

digit3 = 0;

break;

case 4:

digit4 = 0;

break;

}

}

void main() {

unsigned int count = 0;

while (1) {

// Extract individual digits from the count

unsigned char digit1Value = count % 10;

unsigned char digit2Value = (count / 10) % 10;

unsigned char digit3Value = (count / 100) % 10;

unsigned char digit4Value = (count / 1000) % 10;

// Display the digits on the seven-segment displays

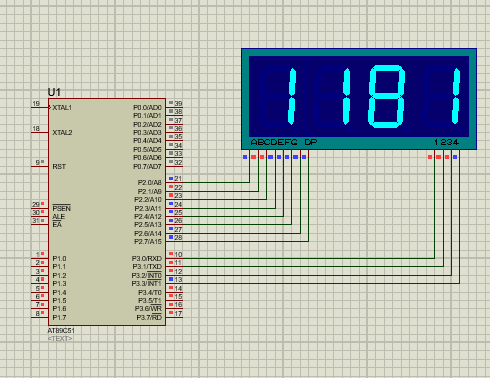
displayDigit(1, digit1Value);

delay(5); // Delay to adjust the brightness

}

}

**Schematric Diagram:**

****