Interfacing of Multiplexed 7-segment display

#include <reg51.h>

void delay(unsigned int time)

{

unsigned int i, j;

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

for (j = 0; j < 1275; j++); // Delay loop

}

void main()

{

unsigned char segment\_code[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90}; // Codes for digits 0-9

unsigned char i;

while (1)

{

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

{

P2 = segment\_code[i]; // Send segment code to P2

delay(500); // Hold the display for some time

}

}

}

Seven segment display with keypad

#include <reg51.h>

// Define ports for keypad and 7-segment display

#define keypad\_port P1

#define seg\_port P2

// Array for displaying digits 0-9 on a common cathode 7-segment display

unsigned char seg\_code[] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80,

0x90};

// Function prototypes

void delay(unsigned int time);

char read\_keypad(void);

void display\_digit(unsigned char digit);

void main() {

char key1, key2, operator;

int num1, num2, result;

while (1) {

// Get first number

key1 = read\_keypad();

num1 = key1 - '0'; // Convert ASCII to integer

display\_digit(num1);

delay(1000);

// Get operator

operator = read\_keypad();

delay(1000);

// Get second number

key2 = read\_keypad();

num2 = key2 - '0'; // Convert ASCII to integer

display\_digit(num2);

delay(1000);

// Perform calculation based on the operator

switch (operator) {

case '+':

result = num1 + num2;

break;

case '-':

result = num1 - num2;

break;

case '\*':

result = num1 \* num2;

break;

case '/':

if (num2 != 0) {

result = num1 / num2;

} else {

result = 0; // Avoid division by zero

}

break;

default:

result = 0;

break;

}

// Display the result on the 7-segment display

display\_digit(result);

delay(2000); // Delay to observe result

}

}

// Function to create delay

void delay(unsigned int time) {

unsigned int i, j;

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

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

}

}

// Function to read pressed key from keypad

char read\_keypad(void) {

while (1) {

keypad\_port = 0xEF; // Test row 1

if (keypad\_port == 0xEE) return '1';

if (keypad\_port == 0xED) return '2';

if (keypad\_port == 0xEB) return '3';

if (keypad\_port == 0xE7) return '+';

keypad\_port = 0xDF; // Test row 2

if (keypad\_port == 0xDE) return '4';

if (keypad\_port == 0xDD) return '5';

if (keypad\_port == 0xDB) return '6';

if (keypad\_port == 0xD7) return '-';

keypad\_port = 0xBF; // Test row 3

if (keypad\_port == 0xBE) return '7';

if (keypad\_port == 0xBD) return '8';

if (keypad\_port == 0xBB) return '9';

if (keypad\_port == 0xB7) return '\*';

keypad\_port = 0x7F; // Test row 4

if (keypad\_port == 0x7E) return 'C';

if (keypad\_port == 0x7D) return '0';

if (keypad\_port == 0x7B) return '=';

if (keypad\_port == 0x77) return '/';

}

}

// Function to display digit on 7-segment display

void display\_digit(unsigned char digit) {

if (digit <= 9) {

seg\_port = seg\_code[digit]; // Display corresponding digit

} else {

seg\_port = 0xFF; // Display nothing for invalid input

}

}

